Overview of GGtools for expression genetics: for 3.7.x

VJ Carey stvjc at channing.harvard.edu

December 14, 2010

Contents

1 Introduction

The *GGtools* package contains infrastructure and demonstration data for joint analysis of transcriptome and genome through combination of DNA expression microarray and high-density SNP genotyping data. For Bioconductor 2.2 we adopted a representation of genotypes due to Clayton (in package *snpMatrix*) allowing reasonably convenient storage and manipulation of 4 megaSNP phase III HapMap genotypes on all the CEPH CEU samples. This contrasts with the previous version of *GGtools* which was limited to 550 kiloSNP and 58 CEU founders.

Note added 2010: GGtools 3.7.x is undergoing revisions to simplify management of the code to face the task of comprehensive eQTL searches. Functions like `gwSnpTests` were created with highly flexible but very complex interface capabilities, and these seem very little used. More important are facilities for concurrent computation and minimum-volume representation of results. The `eqtlTests` function will be the main workhorse for material described in this document; some of the `gwSnpTests` functionalities will be retained for consistency with published work on this package, but the following broad changes to strategy are noted:

- gene symbol translations and gene set manipulation are not addressed in the recommended tools surrounding `eqtlTests`;
- SNP and gene location metadata will not be maintained in the package except for demonstration purposes; location structures will need to be IRanges-based and are the responsibility of the user;

- to cope with volume of test results generated, the `ff` external matrix representation approaches are heavily used.

To give an immediate taste of the capabilities, we attach the package and load some test data.

Originally we used a HapMap3 representation to demonstrate in this vignette, but changes to snpMatrix package made this impractical for development. The devel branch (2.8) of Bioconductor should be consulted, in which snpMatrix2 package is used.

```
> library(GGtools)
> data(hmceuB36.2021)
> hmceuB36.2021

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

Expression data are recoverable in a familiar way:

```
> exprs(hmceuB36.2021)[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
```

Genotype data have more complex representation.

```
> names(smList(hmceuB36.2021))

[1] "20" "21"

> smList(hmceuB36.2021)[1:2]
```

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

$`21`
A snp.matrix with  90 rows and  50165 columns
Row names:  NA06985 ... NA12892
Col names:  rs885550 ... rs10483083
```

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

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

This shows that we use a named list to hold items of the *snp.matrix* class from *snpMatrix*.

It will generally be unnecessary to probe to this level, but it is instructive to check the underlying representation:

```
> schunk = smList(hmceuB36.2021)[["20"]]
> schunk@.Data[1:4, 1:4]
```

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

The leading zeroes show that a raw byte representation is used. We can convert to allele codes as follows:

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

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

[Note mapping to nucleotide codes is not directly supported at this time. The extra baggage is voluminous and for most loci it will never be used. When necessary, manual bioinformatics can be used to check the locus-specific allele frequency in a cohort against the allele count in the data to determine the code unambiguously; the SNP metadata resources in SNPlocs.Hsapiens.* packages, or in ceu1kg, are also relevant.]

The primary analysis of interest is the genome-wide association study, here applied with expression as the phenotype. Here we execute a founders-only analysis, adjusting for gender, confining attention to SNP on chromosome 20. We isolate a single gene, CPNE1, for analysis.

```
> pd = pData(hmceuB36.2021)
> hmFou = hmceuB36.2021[, which(pd$mothid == 0 & pd$fathid == 0)]
> library(illuminaHumanv1.db)
> pc = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
> hmFouc = hmFou[probeId(pc), ]
> hmFouc

snp.matrix-based genotype set:
number of samples:  60
number of chromosomes present:  2
annotation: illuminaHumanv1.db
Expression data dims: 1 x 60
Phenodata: An object of class "AnnotatedDataFrame"
  sampleNames: NA06985 NA06993 ... NA12892 (60 total)
  varLabels: famid persid ... male (7 total)
  varMetadata: labelDescription

> system("rm -rf foo")
> f1 = eqtlTests(hmFouc[chrnum("20"), ], ~male)
> f1

eqtlTools results manager, computed Tue Dec 14 02:53:49 2010
There are 1 chromosomes analyzed.
some genes:  GI_23397697-A
some snps:  rs4814683 rs6076506 ... rs6062370 rs6090120
```

By default, test results are stored in a compact disk-based archive governed by facilities of the *ff* package.

```
> dir("foo")

[1] "foo_20_summ.ff" "foo_chr20.ff"
```

We can get a GRanges instance with the test results for a specific gene:

```
> nn = getNamedLocs(slpack = "SNPlocs.Hsapiens.dbSNP.20100427",
+     chr = "ch20")
> library(GenomicRanges)
> gg = getGRanges(f1, 1, 1, "ch20", nn)
> gg[1:3, ]

GRanges with 3 ranges and 3 elementMetadata values
          seqnames         ranges strand |     score     chisq        df
             <Rle>      <IRanges>  <Rle> | <numeric> <numeric> <numeric>
```

```
rs4814683     ch20 [61795, 61795]      * | 0.1616821      0.16         1
rs6076506     ch20 [63231, 63231]      * | 0.0000000      0.00         1
rs6139074     ch20 [63244, 63244]      * | 0.4683750      0.91         1

seqlengths
 ch20
   NA
```

We convert this to a RangedData instance for viewing on the UCSC genome browser. This code will work in a clean-enough R session; in the sequence of computations given in this vignette it can throw an infinite recursion.

```
> if (interactive()) {
+     s1 = browserSession()
+     rd = RangedData(gg)
+     genome(rd) = "hg18"
+     s1[["CPNE1"]] = rd
+     v1 = browserView(s1, GenomicRanges(33500000, 34500000, "chr20"),
+         track = ucscTrackModes(hide = "knownGene", full = c("refGene",
+             "CPNE1")))
+ }
```

2 Using imputation to 1000 genomes loci

The following code ceased to function after last minute changes to snpMatrix by H-T Leung; to check the behavior of codes related to these, please use R2.13+ and Bioc 2.8+ versions of GGtools.

The *snpMatrix* package defines methods for using linear and haplotype regression to estimate conditional expectations of allele counts for unobserved loci. We used the pilot 1 calls from 1000 genomes (1KG) to define rules relating loci observed in phase III HapMap for CEU to 1KG loci. Here are the first 10 rules for a version of the imputation that has a liberal approach to allowing LD computations.

```
> data(imphm3_1KG_20_mA2)
> imphm3_1KG_20_mA2[1:10]
```

We can use these rules to obtain a richer collection of tests for eQTL for CPNE1:

```
> try(system("rm -rf icpne1"))
> if1 = ieqtlTests(hmFouc[chrnum("chr20"), ], ~male, targdir = "icpne1",
+     runname = "icpne1", rules = imphm3_1KG_20_mA2)
```

The number of tests on the original data:

```
> length(GGtools:::snpIdList(f1)[[1]])
```

and on the imputed data:

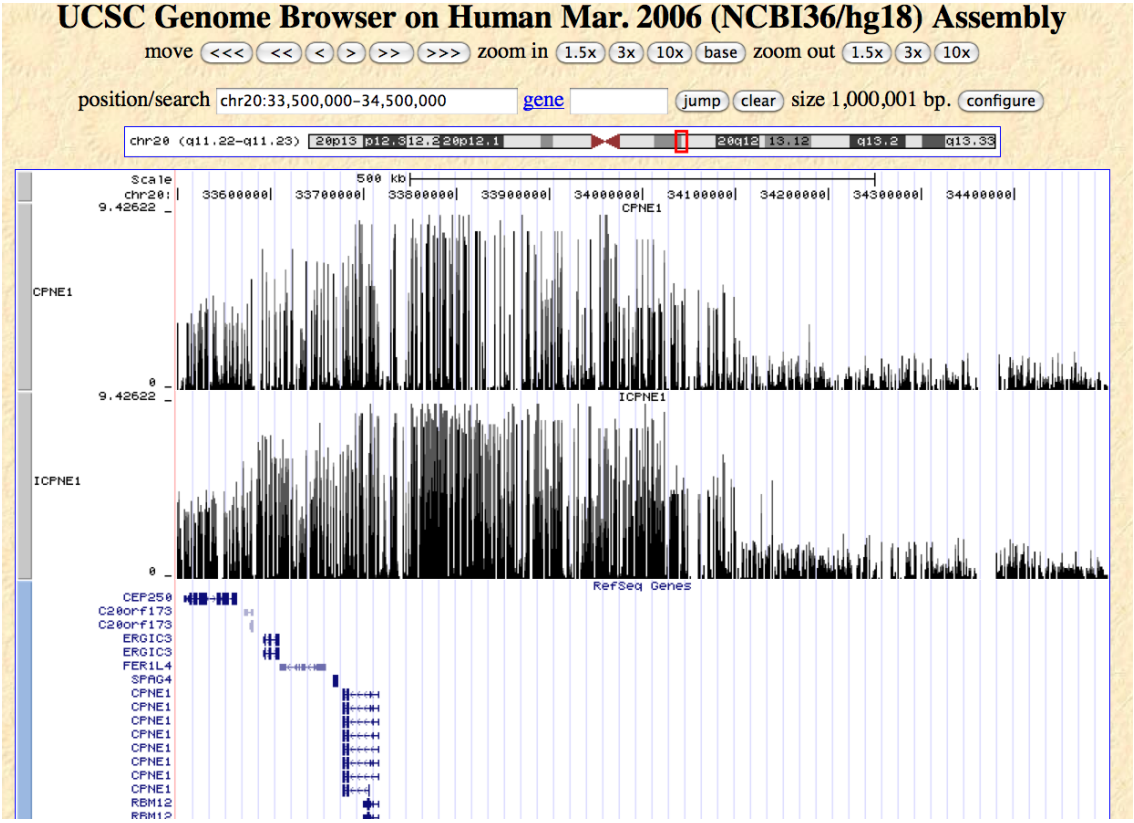
```
> length(GGtools:::snpIdList(if1)[[1]])
```

We need special metadata to get locations for the SNP in this run.

```
> library(ceu1kg)
> data(ceu1kgMeta_20)
> inn = start(ceu1kgMeta_20)
> names(inn) = names(ceu1kgMeta_20)
> igg = getGRanges(if1, 1, 1, "chr20", inn)
```

In the right environment we can use code like the following to see the impact of the imputation.

```
> library(rtracklayer)
> s1 = browserSession()
> genome(rd) = "hg18"
> s1[["CPNE1"]] = rd
> ird = RangedData(igg)
> genome(ird) = "hg18"
> s1[["ICPNE1"]] = ird
> v1 = browserView(s1, GenomicRanges(33.5e6,34.5e6, "chr20"),
+    track=ucscTrackModes(hide="knownGene",
+    full=c("refGene", "CPNE1", "ICPNE1")))
```



We see the expected added density for the richer set of loci, but no qualitative impact of the imputation.

3 Assessing concordance of regression-based imputation with MACH

Again this material cannot be illustrated with snpMatrix; Bioc 2.8 and snpMatrix2 must be used.

Blanca Himes of the Channing Laboratory, Boston, ran the MACH imputation program to compute imputed 'dose' measures based on a certain set of calls for CEU, perhaps HapMap phase III. The `m20` data object for this package includes the results of importing the `mlprob` MACH output for chromosome 20 after imputing to the 1000 genomes loci.

```
> data(m20)
> as(m20, "numeric")[1:10, 1:8]
```

The observed genotypes for CEU founders from HapMap phase III are

```
> c20obs = smList(hmFouc)[[20]]
> c20obs
```

We impute as follows:

```
> unix.time(c20imp <- impute.snps(imphm3_1KG_20_mA2, c20obs))
> c20imp[1:5, 1:5]
```

The results need to be placed in compatible structures. We will use C20D to hold the allele dose measurements from MACH and R20D to hold the snpMatrix regression-based expected allele counts.

```
> C20D = m20[, intersect(colnames(m20), colnames(c20imp))]
> R20D = c20imp[, intersect(colnames(m20), colnames(c20imp))]
> oksnp = intersect(rownames(C20D), rownames(R20D))
> length(oksnp)
> C20D = C20D[oksnp,]
> R20D = R20D[oksnp,]
```

Let's check the basic characteristics of these different imputations. First, how many loci are imputed by MACH?

```
> length(mimp <- setdiff(colnames(m20), colnames(c20obs)))
```

Of these, how many are monomorphic or close to monomorphic?

```
> news = m20[, mimp]
> vn = apply(news, 2, var)
> sum(vn == 0)
> sum(vn < 0.01)
```

The latter result shows that there are over 2000 SNP for which MACH is willing to impute a single het against 109 homozygous individuals. Let's look at an example:

```
> table(as(news[, vn > 0 & vn < 0.01][, 1], "character"))
```

This shows some of the subtlety of the imputations in this case. It would be very nice to reconstruct the imputation for this SNP in detail.

Let's now consider loci imputed by MACH but not by snpMatrix.

```
> if (.Platform$OS.type != "windows") {
+     notreg = colnames(c20imp)[which(apply(c20imp, 2, function(x) all(is.na(x))))]
+     inmachNR = intersect(mimp, notreg)
+     length(inmachNR)
+     m20[1:10, inmachNR[1:5]]
+     table(as(m20[, inmachNR[1]], "character"))
+     table(as(m20[, inmachNR[2]], "character"))
+     library(ceu1kg)
+     if (!exists("ceu1kg.sml"))
```

```
+         data(ceu1kg.sml)
+     rawc20 = ceu1kg.sml[[20]]
+     table(as(rawc20[, "chr20:35790"], "character"))
+     table(as(rawc20[, "chr20:36155"], "character"))
+ }
```

So MACH is able to call 3 hets for each of these loci. We verify that snpMatrix regression does not

```
> if (.Platform$OS.type != "windows") {
+     MbutNR = c("chr20:35790", "chr20:36155")
+     Y = rawc20[, MbutNR, drop = FALSE]
+     X = rawc20[, -match(MbutNR, colnames(rawc20)), drop = FALSE]
+     data(ceu1kgMeta_20)
+     NL = start(ceu1kgMeta_20)
+     names(NL) = names(ceu1kgMeta_20)
+     LOCX = NL[colnames(X)]
+     length(LOCX)
+     LOCY = NL[colnames(Y)]
+     imm = snp.imputation(X, Y, LOCX, LOCY, minA = 2)
+     imm
+     imm1 = snp.imputation(X, Y, LOCX, LOCY, minA = 1)
+     imm1
+ }
```

How liberal are we willing to be with respect to estimation of LD in the presence of very limited genotype variation? The lower we set minA, the more liberal. We can see what the regression procedure does in this case as follows:

```
> if (.Platform$OS.type != "windows") {
+     ii = impute.snps(imm1, rawc20)
+     apply(ii, 2, table)
+ }
```

More uncertainty is evident with the regression imputations, and a very liberal criterion is used to make these calls. It is possible to show that two of the individuals receiving an allele score of .66 with the regression method are called A/B by MACH; the matching between individuals is spotty.