Exploring 1000 Genomes with Bioconductor: Dense SNP Imputation and Sequence-based Expression Genetics

VJ Carey, PhD, Channing Laboratory, Harvard Medical School

July 26, 2010

Contents

1 Introduction

The 1000 genomes (1KG) project aims to develop a "deep catalog of human variation". By employing new techniques for determining the full sequence of individuals' genomes and publishing results at various levels of resolution, the project promises significant data resources for use in the enrichment of theory and methods of statistical genetics. In this paper two applications of resources provided through 1KG are discussed in the context of Bioconductor's R-based facilities for analysis of genome-scale data. First, we discuss methods and performance of imputation of genotypes from relatively sparse SNP panels to the full 1KG panels for CEPH populations. Second, we examine methods for analysis of the genetics of gene expression when next-generation sequencing is used to characterize both DNA variation and gene expression, the latter via the family of methods known as "RNA-seq".

2 SNP imputation to the 1KG panel

2.1 Concept and prevalent approaches

It is widely accepted that genetic association analyses can be enhanced when unobserved genetic markers are suitably imputed for individuals who have only been sparsely genotyped (Marchini et al., 2007; Servin and Stephens, 2007). Imputation schemes have been systematically compared for use in applications, but no clearly dominant method has been identified, using metrics related to accuracy for array-based genotypes compared across array versions (e.g., imputation from the Affymetrix genomewide 5.0 SNP panel to Affymetrix genomewide 6.0 compared to 6.0 calls) or enhancement of power for association studies (Nothnagel et al., 2009). The imputation processes in wide use (based on MACH, BEAGLE, or IMPUTE software) were found in Nothnagel et al. (2009) to require at least hundreds of hours of CPU time with reasonably modern hardware clusters for problems involving imputation of $O(10^6)$ loci on $O(10^3)$ individuals. Nothnagel et al. do not discuss disk space requirements for archiving imputation results, but we have found that storage requirements for using MACH for multiple populations can be substantial. For example, a MACH run imputing to the 1KG SNP panel for 540 individuals generates over 5 gigabytes of text for chromosome 10 alone. This might be regarded as a reasonable fixed cost, with the imputed results serving as a long-term resource. Such a view takes for granted long-term acceptability of both the imputation basis data and the model used along with settings of its tuning parameters. Since the 1KG genotype calls may evolve over time as interpretation of sequencing data and methods for calling improve, and understanding of imputation model performance may also evolve, it is desirable to develop approaches to SNP imputation that are more suited to interactive exploration.

The IDs of samples relevant for these tasks are:

```
> library(ceu1kg)
> data(C20MLDOSE)
> isamp = colnames(C20MLDOSE)
> length(isamp)

[1] 110
```

2.2 Deriving and using SNP imputation rules

2.2.1 A concise representation of the CEU 1KG genotypes

Full 1KG genotype information for 60 individuals from the CEU cohort of the CEPH cell line donors is available in the Bioconductor experimental data package *ceu1kg*. The 1-byte representation of allele copy number (or conditional expectation thereof) defined in the *snpMatrix* package is used (Clayton and Leung, 2007).

```
> library(ceu1kg)  # load all genotype data
> if (!exists("ceu1KG.sml")) data(ceu1KG.sml)
> names(ceu1KG.sml)   # list of autosomes

 [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"  "chr7"  "chr8"  "chr9"
[10] "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
[19] "chr19" "chr20" "chr21" "chr22"

> ceu1KG.sml[[1]]     # representation

A snp.matrix with  60 rows and  605756 columns
Row names:  NA06985 ... NA12874
Col names:  chr1:533 ... chr1:247196267

> object.size(ceu1KG.sml[[1]])

75110112 bytes

> object.size(colnames(ceu1KG.sml[[1]]))  # bookkeeping cost

38760552 bytes

> as(ceu1KG.sml[[1]], "matrix")[1:3,1:4]

        chr1:533 chr1:41342 chr1:41791 chr1:44449
NA06985       01         01         01         01
NA06986       01         02         01         01
NA06994       01         02         01         01

> sum(sapply(ceu1KG.sml, ncol))  # total SNP count

[1] 7724854
```

We'll be working with data on chromosome 20:

```
> c1kg_20 = ceu1KG.sml[[20]]
> dim(c1kg_20)

[1]     60 174484
```

2.2.2 Regression and phasing for deriving imputation rules

1.2 million Phase III HapMap genotypes for 30 CEU trios are available in the Bioconductor *ceuhm3* package. We now describe how to develop imputation rules for loci which were determined to be SNP and were called in 1KG but were not identified as DNA polymorphisms or not genotyped in HapMap Phase III. We will focus on chromosome 20 for illustration.

The imputation procedure provided in the *snpMatrix* package has a reasonably intuitive interface; statistical details will be briefly reviewed below. Complete genotype data on N individuals is managed in an $N \times C$ matrix Z, where C is the complete collection of loci containing all candidates for imputation in new cohorts. The elements of Z are 'B' allele counts and thus take values 0, 1, 2 (only diallelic SNP are handled); a missing indicator is also available for failed genotyping results. Columns of Z is partitioned to form matrices Y and X. The columns of X correspond to loci that have been genotyped in the cohort for which imputation is planned, and the columns of Y correspond to loci that are not genotyped in the cohort for which imputation is planned. Finally, chromosomal addresses for all loci in Y and X are made available, managed in vectors that are denoted l^Y and l^X respectively. Let $t = 1, \ldots, T$ index the columns of Y, and $p = 1, \ldots, P$ index the columns of X. For a fixed value of t we regard the locus defining the tth column of Y as the imputation "target". Rules that use SNPs defining X to predict Y_t are constructed as follows.

- A tuning parameter `try` is selected by the user to specify how many of the SNP in X "nearest" to Y_t (using chromosomal coordinate distance) will be considered as predictors; the default value is 50.

- Three regression tuning parameters are specified by the user: a minimum acceptable value of R^2 to be achieved in stepwise *linear* regression for predicting Y_t using the *try* nearest elements of X; a maximum number of "tagging" SNP to be used in the regression model; and a minimum change in R^2 required to support addition of new SNP to the regression model.

- Two haplotype modeling tuning parameters are specified by the user: a minimum acceptable value of R^2 that must be achieved before resorting to haplotype modeling, and the proportion by which $(1 - R^2)$ must fall relative to the value achieved using regression, in order to adopt the (slower) haplotype-based imputation rule for Y_t.

The details of the regression and haplotype modeling are found in Chapman et al. (2003). Briefly, forward stepwise linear regression is used, governed by the tuning parameters noted above, to establish an initial predictive model. If R^2 is sufficiently large, the model is adopted. Otherwise probabilities of phased haplotypes for the set of predicting and target SNP are computed on the basis of an EM algorithm and a predictive rule is derived from these.

Here we isolate the genotype data for 110 CEU individuals as provided in HapMap Phase III.

```
> library(ceuhm3)
> if (!exists("ceuhm3.sml")) data(ceuhm3.sml)
> hm3_20gt = ceuhm3.sml[[20]][isamp, ]
> hm3_20gt

A snp.matrix with  110 rows and  35305 columns
Row names:  NA06984 ... NA07051
Col names:  rs4814683 ... rs6122298

> dim(hm3_20gt)

[1]   110 35305

> hm3_20snp = colnames(hm3_20gt)
```

The imputation procedure requires information on SNP locations. Such information can be cumbersome to maintain as it is dynamic and voluminous. With Bioconductor, the following steps can be used to obtain locations for SNP from HapMap phase III:

```
> library(SNPlocs.Hsapiens.dbSNP.20090506)
> l20 = getSNPlocs("chr20")
> l20addr = l20$loc
> l20names = paste("rs", l20$RefSNP_id, sep = "")
> names(l20addr) = l20names
```

We need to check if there are SNP in the HapMap data that lack locations:

```
> unloc = setdiff(hm3_20snp, l20names)
> length(unloc)

[1] 110
```

For now, drop these loci

```
> badi = match(unloc, colnames(hm3_20gt))
> hm3_20gt = hm3_20gt[, -badi]
```

and subset the full set of addresses accordingly:

```
> hm3_20locs = l20addr[hm3_20snp]
```

Locations for the 1KG called loci are provided in the *ceu1kg* package:

```
> data(ceu1kgMeta_20)
> ceu1kgMeta_20[1:3, ]

GRanges with 3 ranges and 5 elementMetadata values
           seqnames         ranges strand |         ref         alt       depth
              <Rle>      <IRanges>  <Rle> | <character> <character> <character>
 rs6078030    chr20 [ 9098,  9098]      * |           C           T         192
 rs4814683    chr20 [ 9795,  9795]      * |           G           T         332
rs34147676    chr20 [10731, 10731]      * |           C           A         228
              ancest   alleleCnt
           <character> <character>
 rs6078030           .          24
 rs4814683           .          40
rs34147676           .          14

seqlengths
 chr20
    NA
```

We can make a similar location vector for these loci via

```
> clocs = start(ceu1kgMeta_20)
> names(clocs) = names(ceu1kgMeta_20)
```

and check compatibility of addressing:

```
> inbo = intersect(names(hm3_20locs), names(clocs))
> range(hm3_20locs[inbo] - clocs[inbo])

[1] 0 0
```

Now we proceed to imputation. We first partition the 1KG data into targets `Y` and predictors `X`; predictors are those available in both the HapMap Phase III and 1KG panels; targets are all other 1KG loci.

```
> X = c1kg_20[, intersect(colnames(hm3_20gt), colnames(c1kg_20))]
> Y = c1kg_20[, setdiff(colnames(c1kg_20), colnames(X))]
```

Now impute with two different choices of the `minA` parameter (which determines how much data are available for LD estimation):

```
> options(digits = 3)
> args(snp.imputation)
```

```
function (X, Y, pos.X, pos.Y, phase = FALSE, try = 50, stopping = c(0.95,
    4, 0.05), use.hap = c(0.95, 0.1), em.cntrl = c(50, 0.01),
    minA = 5)
NULL
```

```
> unix.time(imphm3_1KG_20_mA2 <- snp.imputation(X, Y, hm3_20locs[colnames(X)],
+     clocs[colnames(Y)], minA = 2))
```

```
   user  system elapsed
  4.473   0.063   4.656
```

```
> unix.time(imphm3_1KG_20_mA5 <- snp.imputation(X, Y, hm3_20locs[colnames(X)],
+     clocs[colnames(Y)], minA = 5))
```

```
   user  system elapsed
   3.88    0.05    3.93
```

```
> imphm3_1KG_20_mA2[1:5]
```

```
rs6078030  ~  rs6139074 (MAF = 0.2, R-squared = 1)
rs34147676  ~  rs17685809*rs6052070*rs4814683*rs6139074 (MAF = 0.117, R-squared = 0.987
chr20:11541 ~ No imputation available
rs13043000  ~  rs17685809+rs6086539+rs6086616+rs6139074 (MAF = 0.142, R-squared = 0.823
chr20:13532 ~ No imputation available
```

```
> imphm3_1KG_20_mA5[1:5]
```

```
rs6078030  ~  rs6139074 (MAF = 0.2, R-squared = 1)
rs34147676  ~  rs17685809*rs4814683*rs6139074*rs1935386 (MAF = 0.117, R-squared = 0.971
chr20:11541 ~ No imputation available
rs13043000  ~  rs17685809+rs6086539+rs6086616+rs6139074 (MAF = 0.142, R-squared = 0.823
chr20:13532 ~ No imputation available
```

```
> length(imphm3_1KG_20_mA2)
```

```
[1] 141458
```

```
> object.size(imphm3_1KG_20_mA2)
```

```
107483832 bytes
```

```
> summary(imphm3_1KG_20_mA2)
```

```
             SNPs used
R-squared     1 tags (reg) 2 tags (reg) 2 tags (hap) 3 tags (reg) 3 tags (hap)
  (0,0.1]             3193         1228            0            0            0
  (0.1,0.2]              0         3151           19         1179            0
  (0.2,0.3]              0          888          193         1365          141
  (0.3,0.4]              0          422          178          650          405
  (0.4,0.5]              1          295          115          341          325
  (0.5,0.6]              0          271          108          284          271
  (0.6,0.7]              0          232          124          281          305
  (0.7,0.8]              0          491          163          318          292
  (0.8,0.9]              1          821          371          676          616
  (0.9,0.95]             1          946          469          785          834
  (0.95,0.99]         7179         2679          213         1946          741
  (0.99,1]           44554         1108            0         1799          130
  <NA>                   0            0            0            0            0
             SNPs used
R-squared     4 tags (reg) 4 tags (hap)  <NA>
  (0,0.1]                0            0     0
  (0.1,0.2]             57            0     0
  (0.2,0.3]           1167           14     0
  (0.3,0.4]           1387          412     0
  (0.4,0.5]           1049          878     0
  (0.5,0.6]            871         1216     0
  (0.6,0.7]            868         1479     0
  (0.7,0.8]           1202         2078     0
  (0.8,0.9]           1867         3461     0
  (0.9,0.95]          1559         3413     0
  (0.95,0.99]         1387         4154     0
  (0.99,1]             929         2109     0
  <NA>                   0            1 26802
```

```
> save(imphm3_1KG_20_mA2, file = "imphm3_1KG_20_mA2.rda")
> save(imphm3_1KG_20_mA5, file = "imphm3_1KG_20_mA5.rda")
```

The summary shows that about 26000 targets could not be imputed when minA is set to 2, but (summary not shown) there are over 47000 loci not imputed when minA takes default value 5. Which choice is better?

2.3 Comparison to MACH

Blanca Himes of Channing Laboratory generated 1KG imputed genotypes using MACH. The results for chromosome 20 on the hapmap3 calls for CEU are

```
> data(C20MLDOSE)
> C20MLDOSE[11:15, 1:6]

            NA06984 NA06989 NA12340 NA12341 NA12342 NA12343
rs6086616      1.00    2.00       1    1.00    1.00    2.00
rs6039403      1.00    2.00       2    2.00    2.00    2.00
rs17685809     2.00    2.00       2    2.00    2.00    2.00
chr20:21858    1.99    1.98       2    1.97    1.99    1.90
rs6135141      1.00    0.00       1    1.00    1.00    1.00
```

This combines the imputed and observed genotype calls. The imputed ones are:

```
> C20_MACH_FULL = C20MLDOSE
> C20IMP_MACH = C20MLDOSE[setdiff(rownames(C20MLDOSE), colnames(hm3_20gt)),
+     ]
```

The SNP imputed by snpMatrix regression are

```
> C20IMP_REG_SAVE = C20IMP_REG = t(impute.snps(imphm3_1KG_20_mA2,
+     hm3_20gt))
```

The intersection of imputed loci and samples is

```
> imploc = intersect(rownames(C20IMP_MACH), rownames(C20IMP_REG))
> comm = intersect(colnames(C20MLDOSE), colnames(C20IMP_REG))
```

We now have conformant structures on imputed allele counts. We examine correlations for SNP with complete imputations by both methods.

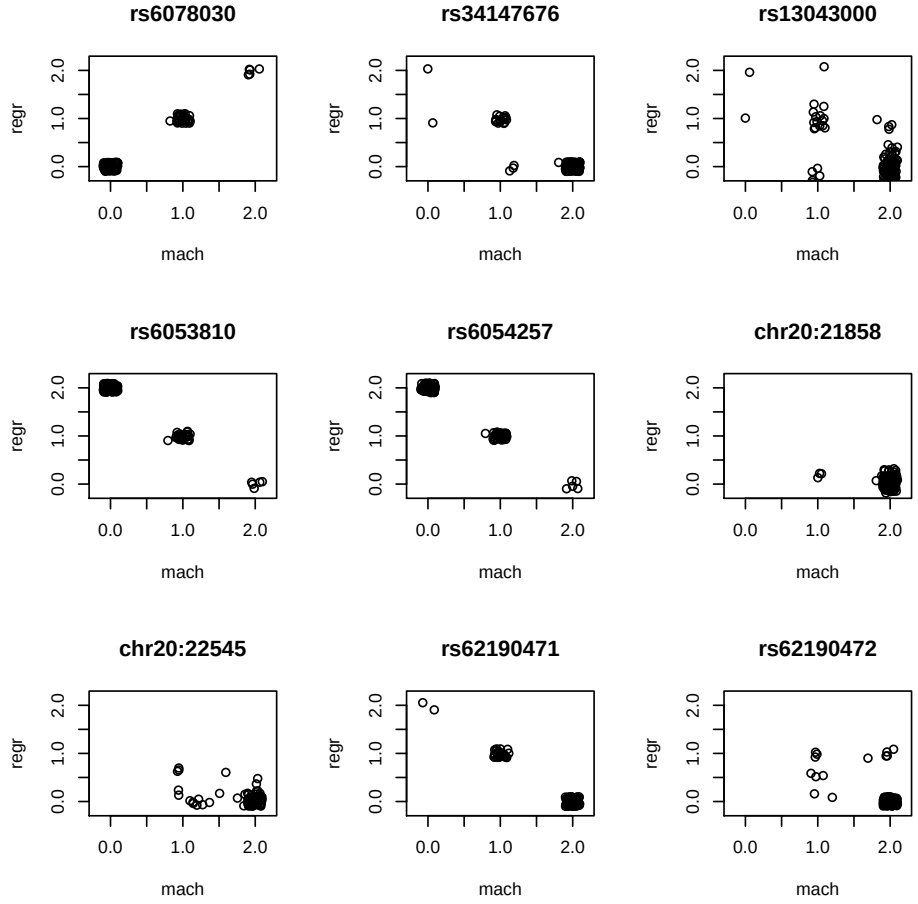
```
> C20IMP_MACH = C20IMP_MACH[imploc, comm]
> C20IMP_REG = C20IMP_REG[imploc, comm]
> regRefuse = apply(C20IMP_REG, 1, function(x) all(is.na(x)))
> regCompleteInds = which(apply(C20IMP_REG, 1, function(x) !any(is.na(x))))
> compcor = sapply(regCompleteInds, function(i) cor(C20IMP_MACH[i,
+     ], C20IMP_REG[i, ]))
> names(compcor) = rownames(C20IMP_REG)[regCompleteInds]
```

We look at absolute value of correlation because we have no guarantees that alleles are identically labeled for the two approaches.

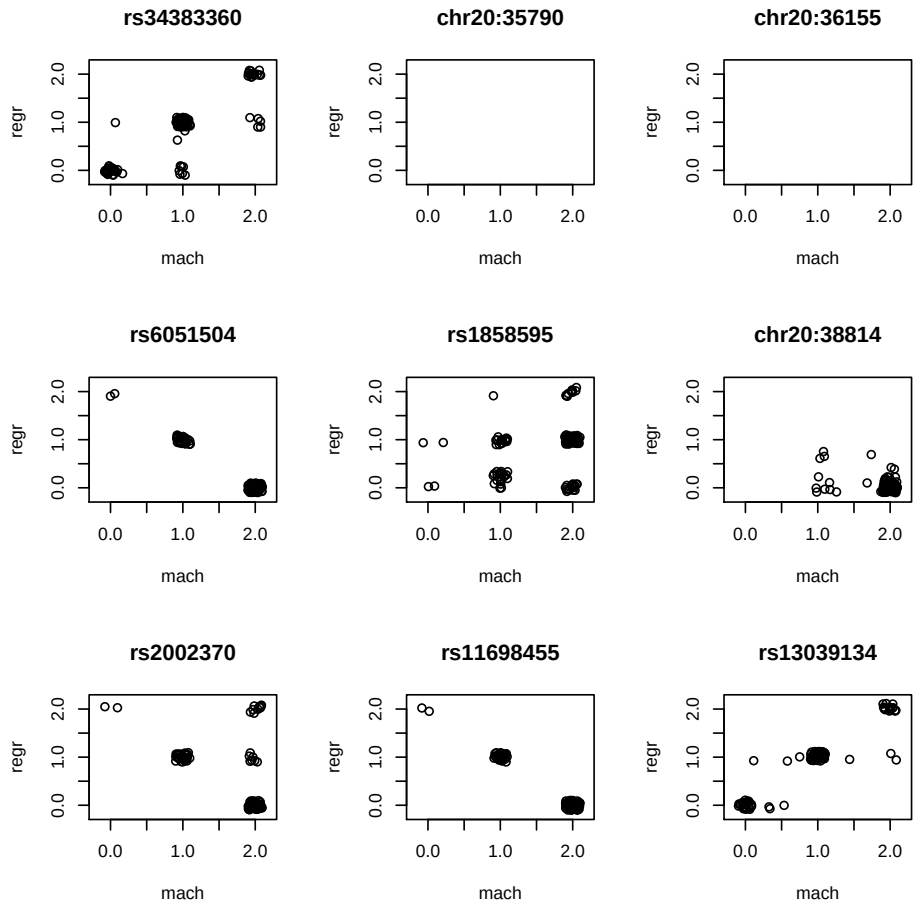
```
> summary(abs(compcor))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.000   0.598   0.974   0.777   1.000   1.000   1.000
```

Here are 9 examples of the correspondence:

```
> par(mfrow = c(3, 3))
> jj = function(x) jitter(x, a = 0.1)
> for (i in 1:9) plot(jj(C20IMP_MACH[i, ]), jj(C20IMP_REG[i, ]),
+     xlab = "mach", ylab = "regr", main = rownames(C20IMP_MACH)[i],
+     xlim = c(-0.2, 2.2), ylim = c(-0.2, 2.2))
```




```
> par(mfrow = c(3, 3))
> for (i in 10:18) plot(jj(C20IMP_MACH[i, ]), jj(C20IMP_REG[i,
+     ]), xlab = "mach", ylab = "regr", main = rownames(C20IMP_MACH)[i],
+     xlim = c(-0.2, 2.2), ylim = c(-0.2, 2.2))
```



There is some concordance and some discordance. It would be interesting to have a metric on imputation effectiveness. MACH and snpMatrix developers have looked at imputation on known loci that were artificially held back and did not discover major shortcomings of their methods. No succinct measure of effectiveness seems forthcoming. MACH has a "QC" measure, and snpMatrix has the R^2 associated with each regression rule.

3 Imputation: effects of different methods on eQTL discovery

3.1 A quick example

We have expression data from HapMap phase III cell lines. In the following we will check a single arbitrarily chosen gene for any eQTL by testing for every SNP, measured or imputed, on chr20, using lm (so a slow solution). We do not require conformity of the SNP sets – the best fit for one imputation method might involve a SNP unavailable

in the other. Samples are restricted primarily by availability of expression data.

```
> library(ceuhm3)
> data(hm3ceuSMS)
> ex3 = exprs(hm3ceuSMS)[, intersect(colnames(C20IMP_MACH), sampleNames(hm3ceuSMS))]
> mach20 = C20MLDOSE[, colnames(ex3)]
> reg20 = C20IMP_REG_SAVE[, colnames(ex3)]
> pv = function(...) {
+     ttmp = try(lm(...))
+     if (inherits(ttmp, "try-error"))
+         return(NA)
+     tmp = summary(ttmp)$coef
+     if (nrow(tmp) != 2)
+         return(NA)
+     return(tmp[2, 4])
+ }
> pv(ex3[1, ] ~ mach20[1, ])

[1] 0.339

> library(multicore)

> try(load("rung1.rda"))
> if (!exists("rung1")) rung1 = unlist(mclapply(1:nrow(mach20),
+     function(x) {
+         if (x%%100 == 0)
+             cat(x)
+         pv(ex3[1, ] ~ mach20[x, ])
+     }))
> save(rung1, file = "rung1.rda")
> try(load("rung2.rda"))
> if (!exists("rung2")) rung2 = unlist(mclapply(1:nrow(reg20),
+     function(x) {
+         if (x%%100 == 0)
+             cat(x)
+         pv(ex3[1, ] ~ reg20[x, ])
+     }))
> save(rung2, file = "rung2.rda")
> names(rung1) = rownames(mach20)
> names(rung2) = rownames(reg20)

> s1 = sort(rung1)[1:20]
> s1
```

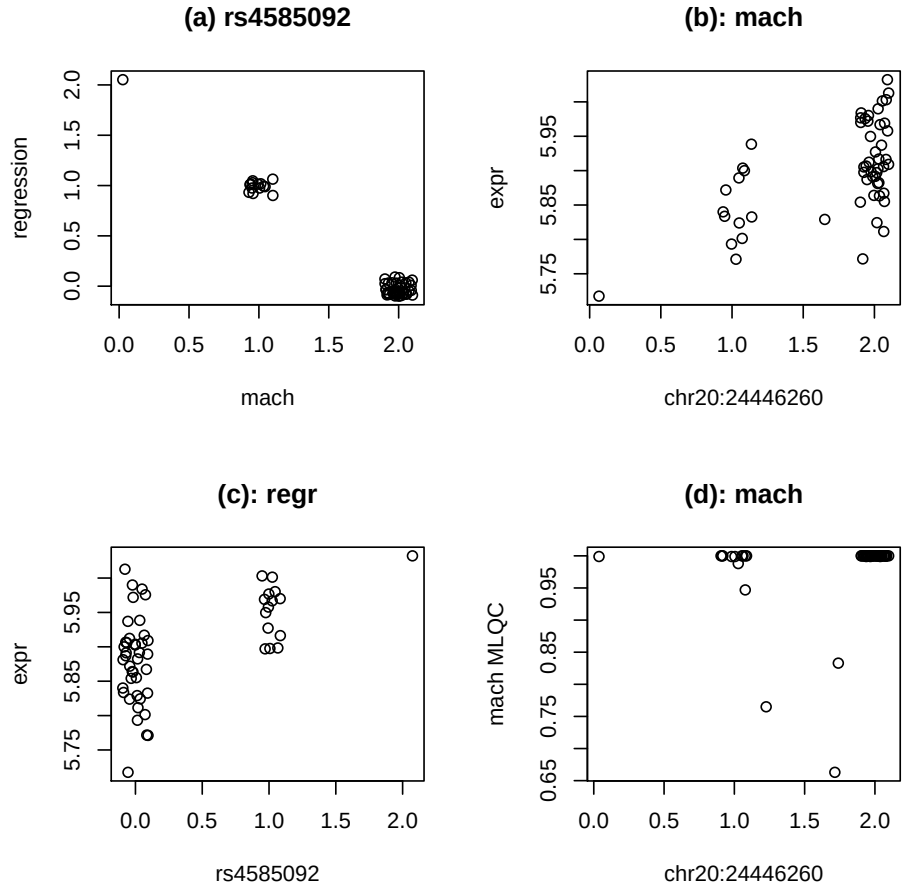
```
chr20:24446260 chr20:24446267      rs4585092      rs2869635 chr20:50314745
      5.11e-06       5.11e-06       1.93e-05       1.93e-05       1.93e-05
chr20:50317701 chr20:59554401      rs6054605     rs57927783      rs6096853
      1.93e-05       4.40e-05       6.72e-05       7.27e-05       7.59e-05
     rs6096856      rs6091494 chr20:50245905 chr20:52190484      rs7268734
      7.59e-05       7.61e-05       7.90e-05       1.19e-04       1.41e-04
chr20:60406403 chr20:61083135      rs6090184 chr20:61088402      rs2427467
      1.45e-04       1.61e-04       1.63e-04       1.63e-04       1.63e-04
```

```
> s2 = sort(rung2)[1:20]
> s2
```

```
     rs4585092      rs6096856      rs6013427      rs3861323     rs34084123
      1.93e-05       1.93e-05       8.16e-05       1.02e-04       1.43e-04
    rs57927783     rs60746501 chr20:24610191 chr20:24610193     rs17226908
      1.43e-04       1.43e-04       1.47e-04       1.47e-04       1.63e-04
chr20:24611003     rs35820407       rs734288     rs12625366     rs12625360
      1.77e-04       1.77e-04       1.77e-04       1.77e-04       1.77e-04
    rs12624535     rs35015656     rs34241321 chr20:24616127     rs35669555
      1.77e-04       1.77e-04       1.77e-04       1.77e-04       1.77e-04
```

We see that there is a better result among the mach-imputed SNP, though it might not be considered genome-wide significant. We will get into significance later. We plot the calls on the best mach-imputed snp in panel (a) below. In panels (b) and (c) we show expression values for the gene used against the SNP identified as most predictive by eQTL analysis based on MACH imputations and regression imputation respectively. The best identified SNP for MACH isolates an A/A with relatively low expression; the best identified SNP with regression isolates an A/A with relatively high expression. The findings are almost certainly false positives, but they indicate some of the complexity of interpretation that can crop up.

```
> par(mfrow = c(2, 2))
> besti = intersect(names(s1), names(s2))[1]
> bestm = names(s1)[1]
> bestr = names(s2)[1]
> plot(jj(mach20[besti, ]), jj(reg20[besti, ]), xlab = "mach",
+     ylab = "regression", main = paste("(a)", besti))
> plot(ex3[1, ] ~ jj(mach20[bestm, ]), xlab = bestm, ylab = "expr",
+     main = "(b): mach")
> plot(ex3[1, ] ~ jj(reg20[bestr, ]), xlab = bestr, ylab = "expr",
+     main = "(c): regr")
> data(C20MLQC)
> plot(C20MLQC[bestm, colnames(mach20)] ~ jj(mach20[bestm, ]),
+     xlab = bestm, ylab = "mach MLQC", main = "(d): mach")
```



It is not clear how to form a preference for one approach to imputing genotypes at this locus. Here is the model chosen by snpMatrix:

```
> imphm3_1KG_20_mA2[bestr]

rs4585092  ~  rs2869635 (MAF = 0.142, R-squared = 1)
```

3.2 A screen of cis relationships on chromosome 20

3.2.1 Checking concordance between Mach and regression imputation for an apparently SNP-regulated gene

We have a high performance approach to surveying all genes vs all SNP on chromosome 20, with regression imputation.

```
> library(GGtools)

Loading package ff 2.1-2
- getOption("fftempdir")=="/var/folders/4D/4DI98FkjGzq0K2niUTEHSE+++TM/-Tmp-//RtmpJx5bQ
```

```
- getOption("ffextension")=="ff"
- getOption("ffdrop")==TRUE
- getOption("fffinonexit")==TRUE
- getOption("ffpagesize")==65536
- getOption("ffcaching")=="mmnoflush"  -- consider "ffeachflush" if your system stalls
- getOption("ffbatchbytes")==16777216 -- consider a different value for tuning your sys
Attaching package ff
```

```
> h20 = hm3ceuSMS[chrnum("chr20"), ]
> library(illuminaHumanv1.db)
> g20 = get("20", revmap(illuminaHumanv1CHR))
> h20 = h20[probeId(g20), ]
> if (!exists("ieq20")) {
+     if (!file.exists("ieq20.rda"))
+         ieq20 = ieqtlTests(h20, ~male, runname = "iq20", targdir = "iq20",
+             rules = imphm3_1KG_20_mA2, geneApply = mclapply)
+     else load("ieq20.rda")
+ }
> ieq20

eqtlTools results manager, computed Sun Jul 25 21:51:15 2010
There are 1 chromosomes analyzed.
some genes:  GI_4557248-S GI_15451784-S ... GI_42662261-S GI_18592500-S
some snps:  rs4814683 rs6139074 ... rs4057205 rs7270744
```

We will now obtain the best 10 SNP for each gene:

```
> if (!exists("s20tops")) s20topsL = lapply(g20, function(x) topFeats(probeid = x,
+     mgr = ieq20, ffind = 1, anno = "illuminaHumanv1.db", useSym = FALSE))
> names(s20topsL) = g20
> cls20topsL = s20topsL
> names(cls20topsL) = NULL
> s20tops = unlist(lapply(cls20topsL, function(x) {
+     nn = names(x)
+     tmp = as.numeric(x[1])
+     names(tmp) = nn[1]
+     tmp
+ }))
> topsnp20 = names(s20tops)
> tope20df = data.frame(snpid = topsnp20, score = s20tops, gene = g20)
```

and look at the genes that show strong association with some SNP.

```
> tops = tope20df[order(tope20df$score, decreasing = TRUE), ][1:10,
+     ]
> tops
```

```
             snpid score          gene
342     rs16987681  65.0 GI_18426907-I
174      rs6060632  49.1 GI_23397697-A
604      rs1739652  39.9   Hs.400876-S
204     rs62205578  37.6 GI_18379355-I
258      rs2297154  34.7 GI_33356545-A
192     rs17123741  29.9 GI_31652256-S
514 chr20:61206271  28.4 GI_29826314-I
460      rs1776953  27.9 GI_14042959-S
56       rs2746621  27.6 GI_30581163-S
254 chr20:59719605  27.3 GI_37059782-S
```

```
> bestg = as.character(tops[1, "gene"])
> theg = get(bestg, illuminaHumanv1SYMBOL)
> theg
```

```
[1] "SIRPG"
```

Does imputation with MACH lead to different identifications of eQTL for these genes? Here's the first one

```
> try(load("runb1.rda"))
> if (!exists("runb1")) {
+     unix.time(runb1 <- unlist(mclapply(1:nrow(C20MLDOSE), function(x) {
+         if (x%%1000 == 0)
+             cat(x)
+         pv(ex3[bestg, ] ~ C20MLDOSE[x, intersect(colnames(C20MLDOSE),
+             colnames(ex3))][colnames(ex3)])
+     })))
+     names(runb1) = rownames(C20MLDOSE)
+     save(runb1, file = "runb1.rda")
+ }
```

The top 10 SNP for predicting SIRPG using MACH imputation are

```
> topm = sort(runb1)[1:10]
> topm
```

```
chr20:43126848 chr20:54728042 chr20:43495672 chr20:55076615      rs1853154
     2.10e-23       2.17e-23       2.26e-23       2.33e-23       2.38e-23
 chr20:8268262 chr20:58944133 chr20:57633982      rs6055751 chr20:22987542
     2.38e-23       2.38e-23       2.38e-23       2.38e-23       2.38e-23
```

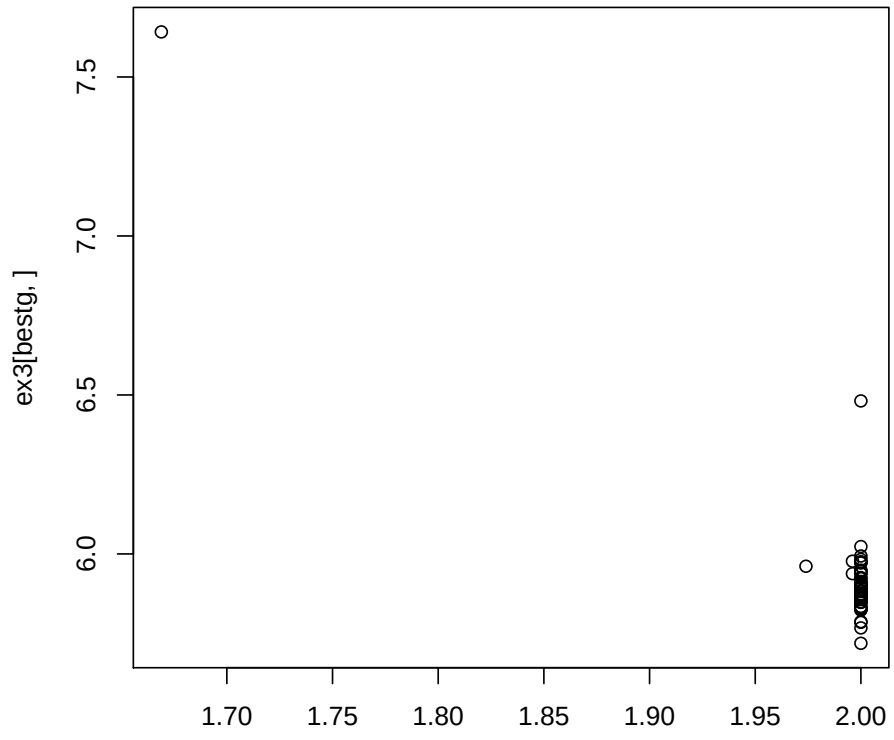
those based on regression imputation are

```
> topr = sort(s20topsL[[bestg]], decreasing = TRUE)[1:10]
> topr
```

```
    rs16987681     rs11906667      rs8118004      rs7268801      rs2223275
          65.0           47.4           47.4           47.4           47.4
chr20:10153879 chr20:10162038 chr20:35460299      rs2294565 chr20:50121440
          47.4           47.4           47.4           47.4           47.4
```

The mach-based result is probably not very interesting.

```
> plot(ex3[bestg, ] ~ C20MLDOSE[names(topm)[1], intersect(colnames(C20MLDOSE),
+     colnames(ex3))])
```

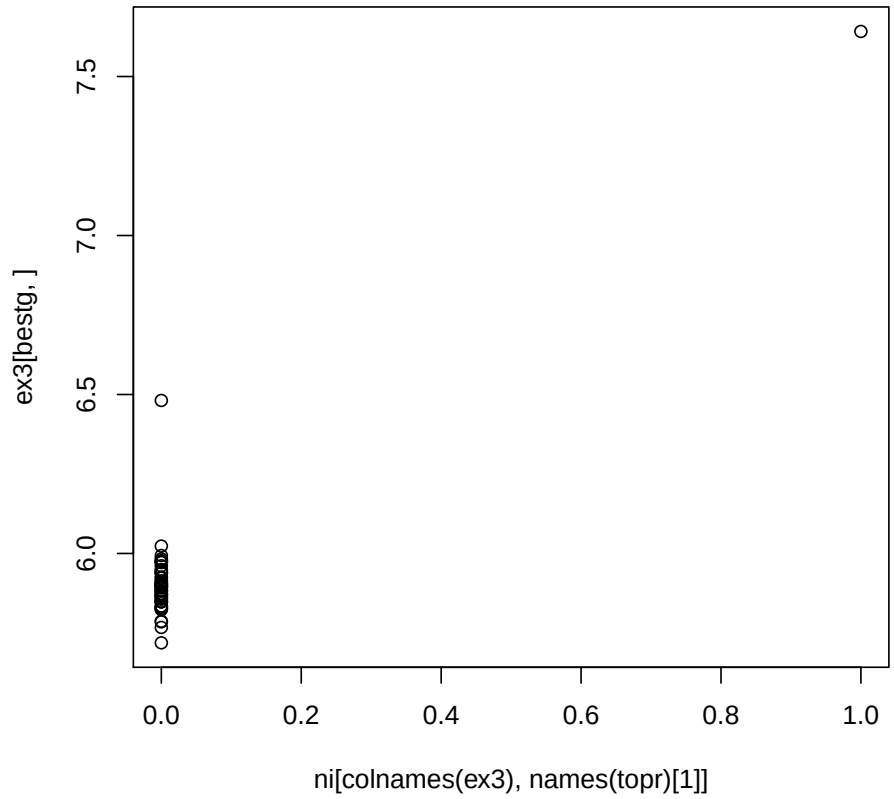


To get a similar plot with the on-demand regression imputation, we have to create the imputed data and extract:

```
> sss = smList(h20)[[1]]
> ni = as(sss, "numeric")
> ni[1:5, 1:5]
```

```
        rs4814683 rs6139074 rs1418258 rs6086616 rs6039403
NA12146         2         1         2         0         2
NA12239         0         0         0         0         2
NA12145         1         1         1         0         2
NA10847         1         0         1         0         2
NA12144         1         1         1         1         2
```

```
> plot(ex3[bestg, ] ~ ni[colnames(ex3), names(topr)[1]])
```



We see that both of these apparently strong results are based on a single heterozygous individual (in the case of mach, one with 1.7 B alleles, say.)

Clearly the use of extreme statistics has to be accompanied by additional assessment.

3.2.2 Mach-based tests for other genes

References

Juliet M Chapman, Jason D Cooper, John A Todd, and David G Clayton. Detecting disease associations due to linkage disequilibrium using haplotype tags: a class of tests and the determinants of statistical power. *Hum Hered*, 56(1-3):18–31, Jan 2003. doi: 10.1159/000073729.

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

Jonathan Marchini, Bryan Howie, Simon Myers, Gil McVean, and Peter Donnelly. A new multipoint method for genome-wide association studies by imputation of genotypes. *Nat Genet*, 39(7):906–13, Jul 2007. doi: 10.1038/ng2088. URL `http://www.nature.com/ng/journal/v39/n7/abs/ng2088.html`.

Michael Nothnagel, David Ellinghaus, Stefan Schreiber, Michael Krawczak, and Andre Franke. A comprehensive evaluation of snp genotype imputation. *Hum Genet*, 125 (2):163–71, Mar 2009. doi: 10.1007/s00439-008-0606-5.

Bertrand Servin and Matthew Stephens. Imputation-based analysis of association studies: candidate regions and quantitative traits. *PLoS Genet*, 3(7):e114, Jul 2007. doi: 10.1371/journal.pgen.0030114. URL `http://www.plosgenetics.org/article/info%253Adoi%252F10.1371%252Fjournal.pgen.0030114`.