snpMatrix vignette
Example of genome-wide association testing

David Clayton and Chris Wallace

October 24, 2025

The `snpMatrix` package

The package "`snpMatrix`" was written to provide data classes and methods to facilitate the analysis of whole genome association studies in R. In the data classes it implements, each genotype call is stored as a single byte and, at this density, data for single chromosomes derived from large studies and new high-throughput gene chip platforms can be handled in memory by modern PCs and workstations. The object–oriented programming model introduced with version 4 of the S-plus package, usually termed "S4 methods" was used to implement these classes.

At the current state of development the package only supports population–based studies, although we would hope to provide support for family–based studies soon. Both quantitative and qualitative phenotypes may be analysed. Flexible association testing functions are provided which can carry out single SNP tests which control for potential confounding by quantitative and qualitative covariates. Tests involving several SNPs taken together as "tags" are also supported. Efficient calculation of pair-wise linkage disequilibrium measures is implemented and data input functions include a function which can download data directly from the international HapMap project website.

The package is described by Clayton and Leung (2007) *Human Heredity*, **64**: 45–51.

Getting started

We shall start by loading the the packages and the data to be used in this exercise:

```
> library(chopsticks)
> library(hexbin)
> data(for.exercise)
```

In addition to the `snpMatrix` package, we have also loaded the `hexbin` package which reduces file sizes and legibility of plots with very many data points.

The data have been created artificially from publicly available datasets. The SNPs have been selected from those genotyped by the International HapMap Project[1] to represent the typical density found on a whole genome association chip, (the Affymetrix 500K

[1] `http://www.hapmap.org`

platform[2]) for a moderately sized chromosome (chromosome 10). A (rather too) small study of 500 cases and 500 controls has been simulated allowing for recombination using beta software from Su and Marchini. Re-sampling of cases was weighted in such a way as to simulate three "causal" locus on this chromosome, with multiplicative effects of 1.3, 1.4 and 1.5 for each copy of the risk allele at each locus. It should be noted that this is a somewhat optimistic scenario!

You have loaded three objects:

1. `snps.10`, an object of class "`snp.matrix`" containing a matrix of SNP genotype calls. Rows of the matrix correspond to subjects and columns correspond to SNPs:

   ```
   > show(snps.10)

   A snp.matrix with  1000 rows and  28501 columns
   Row names:  jpt.869 ... ceu.464
   Col names:  rs7909677 ... rs12218790
   ```

2. `snp.support`, a conventional R data frame containing information about the SNPs typed. To see its contents:

   ```
   > summary(snp.support)

     chromosome    position          A1        A2
   Min.   :10   Min.   :   101955   A:14019   C: 2349
   1st Qu.:10   1st Qu.: 28981867   C:12166   G:12254
   Median :10   Median : 67409719   G: 2316   T:13898
   Mean   :10   Mean   : 66874497
   3rd Qu.:10   3rd Qu.:101966491
   Max.   :10   Max.   :135323432
   ```

   Row names of this data frame correspond with column names of `snps.10` and comprise the (unique) SNP identifiers.

3. `subject.support`, another conventional R data frame containing further information about the subjects. The row names coincide with the row names of `snps.10` and comprise the (unique) subject identifiers. In this simulated study there are only two variables:

   ```
   > summary(subject.support)

         cc          stratum
   Min.   :0.0   CEU    :494
   1st Qu.:0.0   JPT+CHB:506
   Median :0.5
   Mean   :0.5
   3rd Qu.:1.0
   Max.   :1.0
   ```

[2] `http://www.affymetrix.com/support/technical/sample_data/500k_hapmap_genotype_data.affx`

The variable `cc` identifies cases (`cc=1`) and controls (`cc=0`) while `stratum`, coded 1 or 2, identifies a stratification of the study population — more on this later.

In general, analysis of a whole–genome association study will require a subject support data frame, a SNP support data frame for each chromosome, and a SNP data file for each chromosome[3].

You may have noticed that it was not suggested that you should examine the contents of `snps.10` by typing `summary(snps.10)`. The reason is that this command produces one line of summary statistics for each of the 12,400 SNPs. Instead we shall compute the summary and then summarise it!

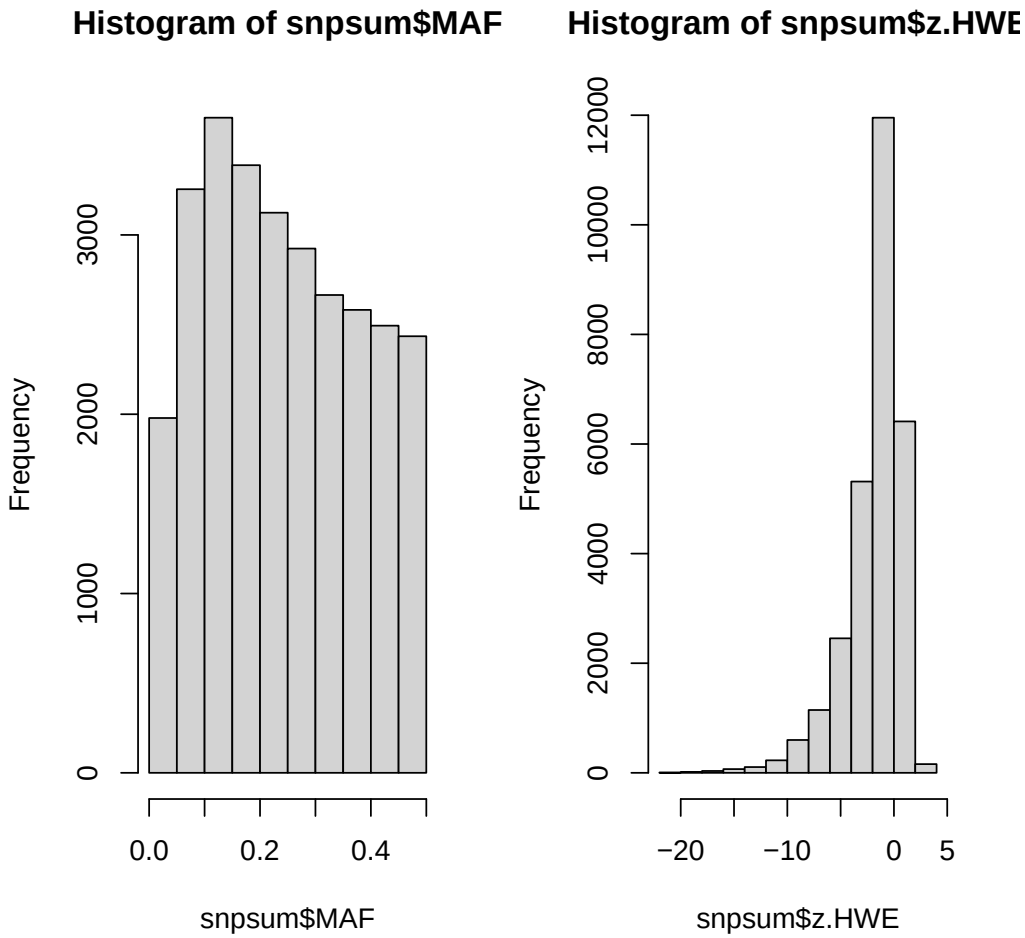
```
> snpsum <- summary(snps.10)
> summary(snpsum)
```

```
     Calls        Call.rate          MAF              P.AA
 Min.   : 975   Min.   :0.975   Min.   :0.0000   Min.   :0.00000
 1st Qu.: 988   1st Qu.:0.988   1st Qu.:0.1258   1st Qu.:0.06559
 Median : 990   Median :0.990   Median :0.2315   Median :0.26876
 Mean   : 990   Mean   :0.990   Mean   :0.2424   Mean   :0.34617
 3rd Qu.: 992   3rd Qu.:0.992   3rd Qu.:0.3576   3rd Qu.:0.60588
 Max.   :1000   Max.   :1.000   Max.   :0.5000   Max.   :1.00000

      P.AB             P.BB             z.HWE
 Min.   :0.0000   Min.   :0.00000   Min.   :-21.9725
 1st Qu.:0.2080   1st Qu.:0.06465   1st Qu.: -2.8499
 Median :0.3198   Median :0.27492   Median : -1.1910
 Mean   :0.3074   Mean   :0.34647   Mean   : -1.8610
 3rd Qu.:0.4219   3rd Qu.:0.60362   3rd Qu.: -0.1014
 Max.   :0.5504   Max.   :1.00000   Max.   :  3.7085
                                    NA's   :4
```

The contents of `snpsum` are fairly obvious from the output from the last command. We could look at a couple of summary statistics in more detail:

```
> par(mfrow = c(1, 2))
> hist(snpsum$MAF)
> hist(snpsum$z.HWE)
```

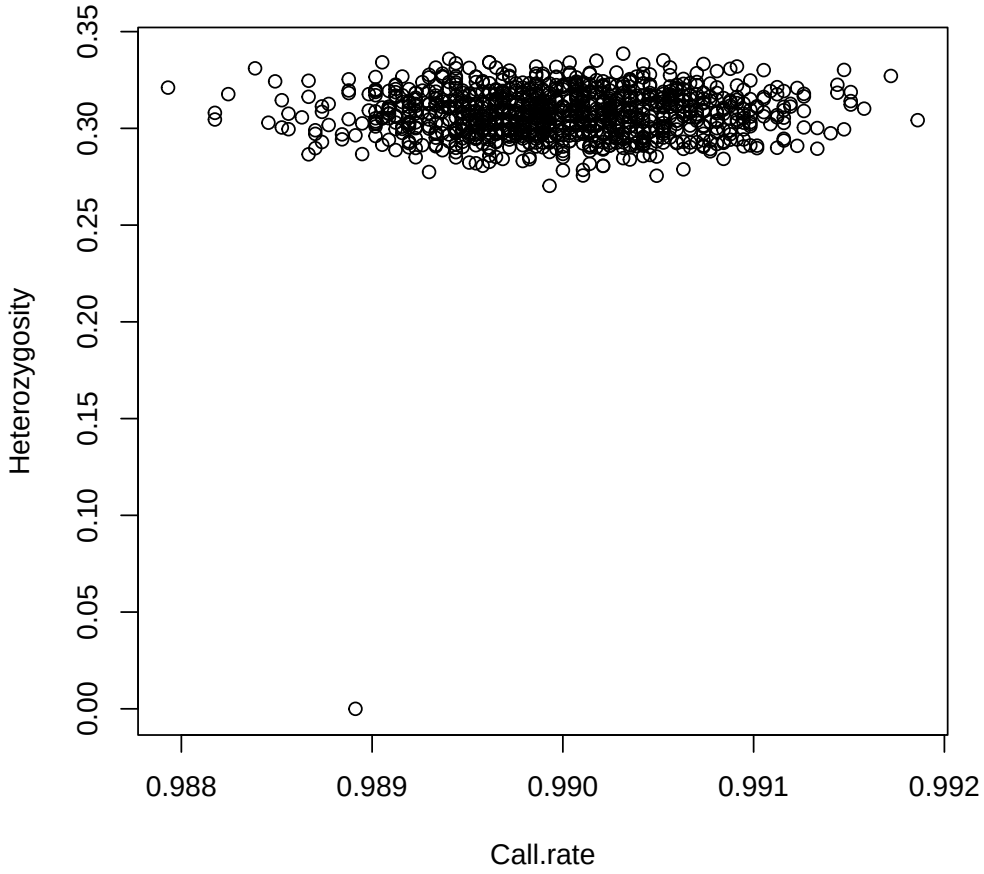
[3] Support files are usually read in with general tools such as `read.table()`. The `snpMatrix` package contains a number of tools for reading SNP genotype data into an object of class "`snp.matrix`".



The latter should represent a z-statistic. *i.e.* a statistic normally distributed with mean zero and unit standard deviation under the hypothesis of Hardy–Weinberg equilibrium (HWE). Quite clearly there is extreme deviation from HWE, but this can be accounted for by the manner in which this synthetic dataset was created.

A useful tool to detect samples that have genotyped poorly is `row.summary()`. This calculates call rate and mean heterozygosity across all SNPs for each subject in turn:

```
> sample.qc <- row.summary(snps.10)
> summary(sample.qc)

   Call.rate      Heterozygosity
 Min.   :0.9879   Min.   :0.0000
 1st Qu.:0.9896   1st Qu.:0.2993
 Median :0.9900   Median :0.3078
 Mean   :0.9900   Mean   :0.3074
 3rd Qu.:0.9904   3rd Qu.:0.3159
 Max.   :0.9919   Max.   :0.3386

> par(mfrow = c(1, 1))
> plot(sample.qc)
```



The analysis

We'll start by removing the three 'outlying' samples above (the 3 samples with Heterozygosity near zero):

```
> use <- sample.qc$Heterozygosity > 0
> snps.10 <- snps.10[use, ]
> subject.support <- subject.support[use, ]
```

Then we'll see if there is any difference between call rates for cases and controls. First generate logical arrays for selecting out cases or controls:[4]

```
> if.case <- subject.support$cc == 1
> if.control <- subject.support$cc == 0
```

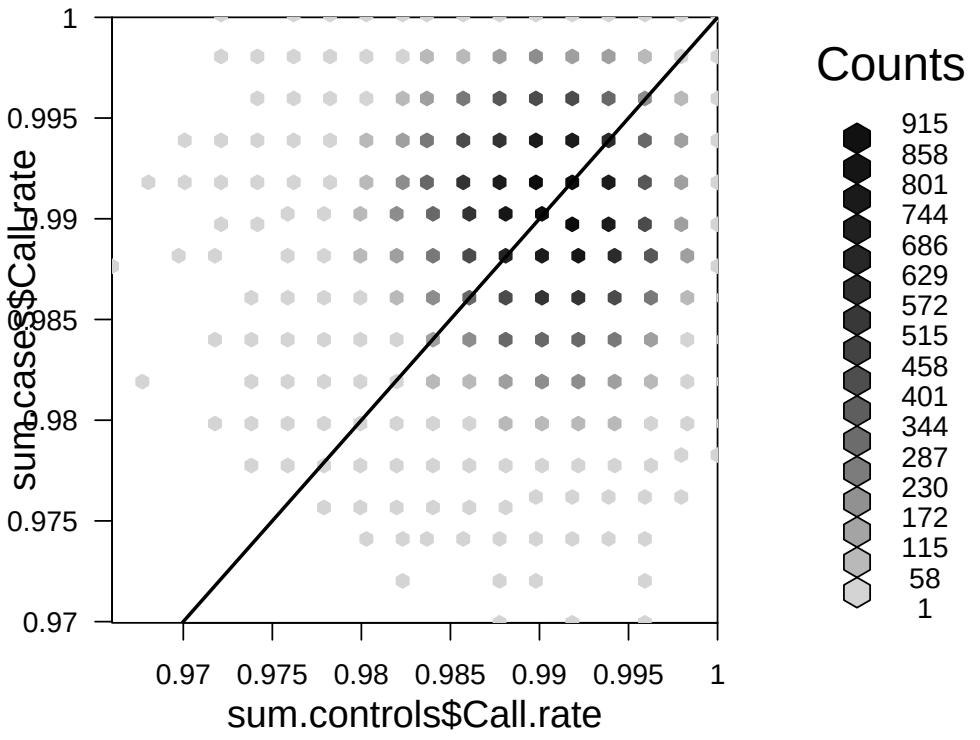
Now we recompute the genotype summary separately for cases and controls:

```
> sum.cases <- summary(snps.10[if.case, ])
> sum.controls <- summary(snps.10[if.control, ])
```

[4]These commands assume that the subject support frame has the same number of rows as the SNP matrix and that they are in the same order. Otherwise a slightly more complicated derivation is necessary.

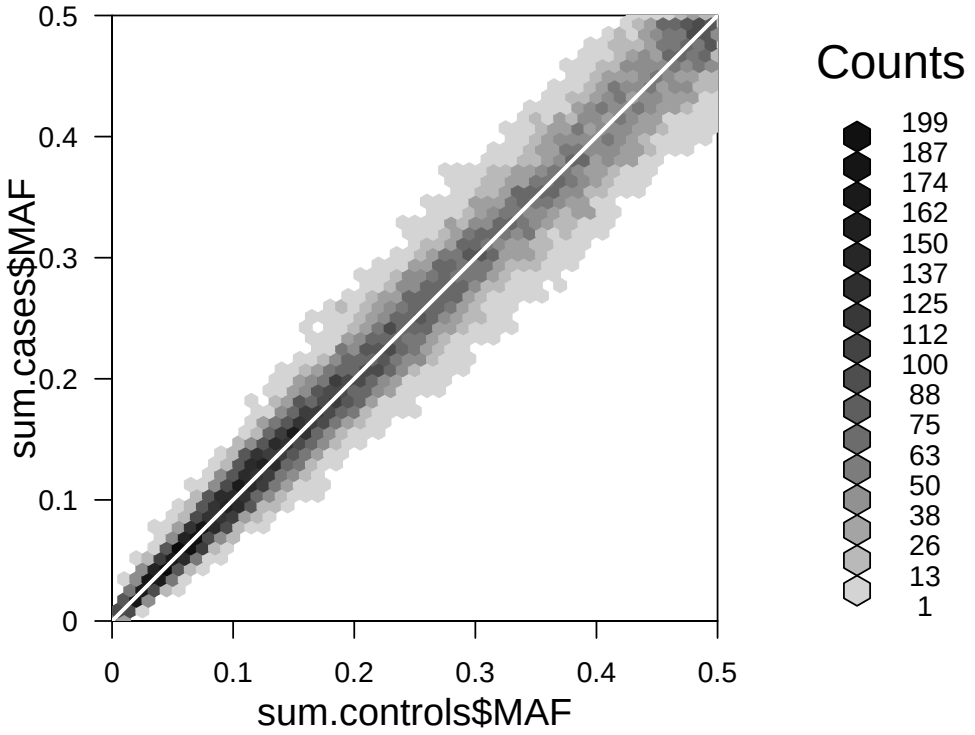
and plot the call rates, using hexagonal binning and superimposing a line of slope 1 through the origin:

```
> hb <- hexbin(sum.controls$Call.rate, sum.cases$Call.rate, xbin = 50)
> sp <- plot(hb)
> hexVP.abline(sp$plot.vp, a = 0, b = 1, col = "black")
```



There is no obvious difference in call rates. This is not a surprise, since no such difference was built into the simulation. In the same way we could look for differences between allele frequencies, superimposing a line of slope 1 through the origin:

```
> sp <- plot(hexbin(sum.controls$MAF, sum.cases$MAF, xbin = 50))
> hexVP.abline(sp$plot.vp, a = 0, b = 1, col = "white")
```



This is not a very effective way to look for associations, but if the SNP calling algorithm has been run separately for cases and controls this plot can be a useful diagnostic for things going wrong (*e.g.* different labelling of clusters).

It should be stressed that, for real data, the plots described above would usually have many more outliers. Our simulation did not model the various biases and genotype failures that affect real studies.

The fastest tool for carrying out simple tests for association taking the SNP one at a time is `single.snp.tests`. The output from this function is a data frame with one line of data for each SNP. Running this in our data and summarising the results:

```
> tests <- single.snp.tests(cc, data = subject.support, snp.data = snps.10)
```

Some words of explanation are required. In the call, the `snp.data=` argument is mandatory and provides the name of the matrix providing the genotype data. The `data=` argument gives the name of the data frame that contains the remaining arguments — usually the subject support data frame[5].

Let us now see what has been calculated:

```
> summary(tests)
```

[5]This is not mandatory — we could have made `cc` available in the global environment. However we would then have to be careful that the values are in the right order; by specifying the data frame, order is forced to be correct by checking the order of the row names for the `data` and `snp.data` arguments.

```
    chi2.1df          chi2.2df          p.1df            p.2df
 Min.   : 0.0000   Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.: 0.1724   1st Qu.: 0.7915   1st Qu.:0.1410   1st Qu.:0.1601
 Median : 0.7729   Median : 1.8562   Median :0.3793   Median :0.3953
 Mean   : 1.5608   Mean   : 2.5961   Mean   :0.4192   Mean   :0.4281
 3rd Qu.: 2.1670   3rd Qu.: 3.6635   3rd Qu.:0.6780   3rd Qu.:0.6732
 Max.   :34.0217   Max.   :37.2487   Max.   :1.0000   Max.   :1.0000
 NA's   :4         NA's   :826       NA's   :4        NA's   :826
       N
 Min.   :974
 1st Qu.:987
 Median :989
 Mean   :989
 3rd Qu.:991
 Max.   :999
```

We have, for each SNP, chi-squared tests on 1 and 2 degrees of freedom (df), together with N, the number of subjects for whom data were available. The 1 df test is the familiar Cochran-Armitage test for codominant effect while the 2 df test is the conventional Pearsonian test for the 3×2 contingency table. The large number of `NA` values for the latter test reflects the fact that, for these SNPs, the minor allele frequency was such that one homozygous genotype did not occur in the data.

We will probably wish to restrict our attention to SNPs that pass certain criteria. For example

```
> use <- snpsum$MAF > 0.01 & snpsum$z.HWE^2 < 200
```

(The Hardy-Weinberg filter is ridiculous and reflects the strange characteristics of these simulated data. In real life you might want to use something like 16, equivalent to a 4SE cut-off). To see how many SNPs pass this filter

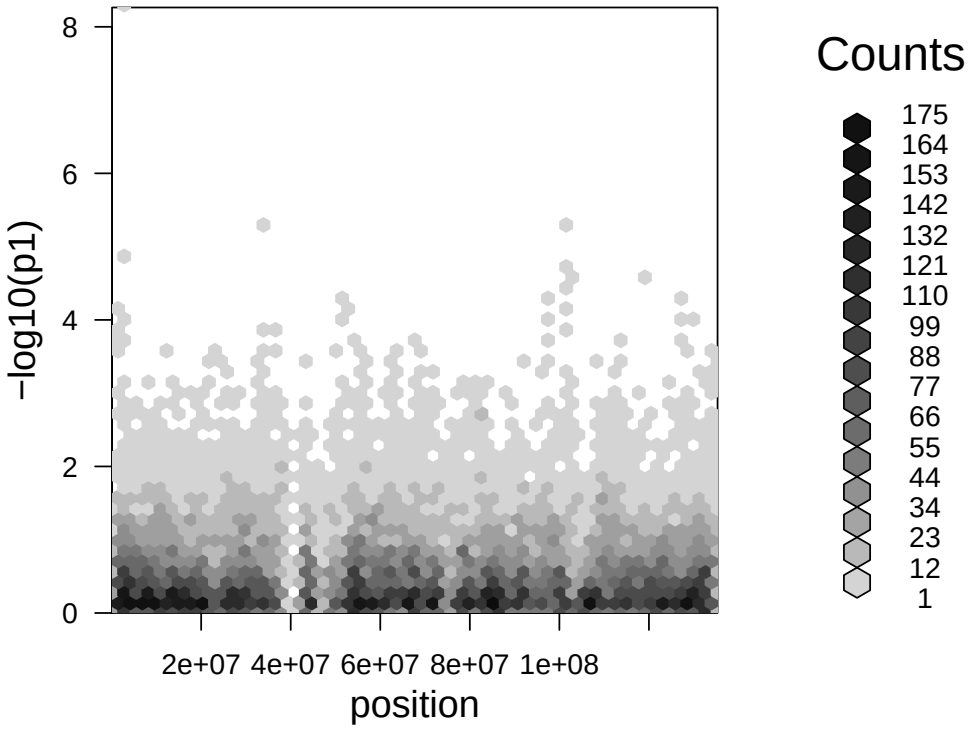
```
> sum(use)

[1] 28184
```

We will now throw way the discarded test results and save the positions of the remaining SNPs

```
> tests <- tests[use, ]
> position <- snp.support[use, "position"]
```

We now calculate p-values for the Cochran-Armitage tests and plot minus logs (base 10) of the p-values against position, with a horizontal line corresponding to $p = 10^{-6}$:

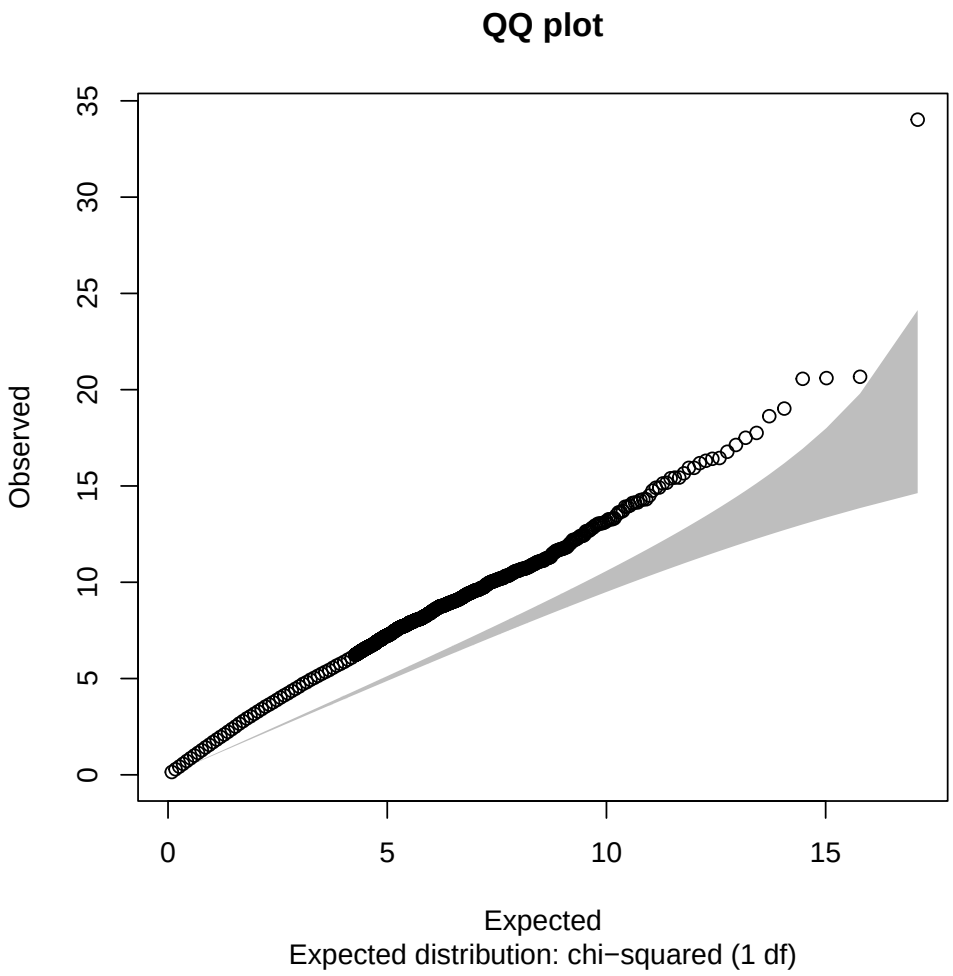
```
> p1 <- pchisq(tests$chi2.1df, df = 1, lower.tail = FALSE)
> plot(hexbin(position, -log10(p1), xbin = 50))
```



Clearly there are far too many "significant" results, an impression which is made even clearer by the quantile-quantile (QQ) plot:

```
> qq.chisq(tests$chi2.1df, df = 1)
```

```
           N      omitted       lambda
28184.000000     0.000000     1.676657
```

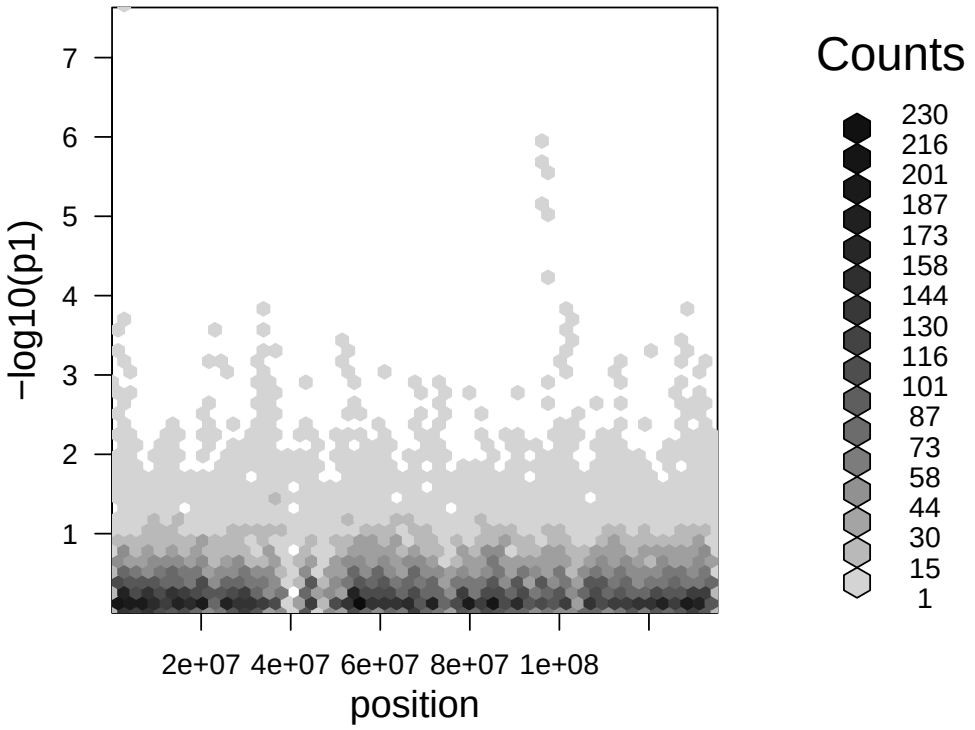


The three numbers returned by this command are the number of tests considered, the number of outliers falling beyond the plot boundary, and the slope of a line fitted to the smallest 90% of values. The "concentration band" for the plot is shown in grey. This region is defined by upper and lower probability bounds for each order statistic. The default is to use the 2.5% and 95.7% bounds[6].

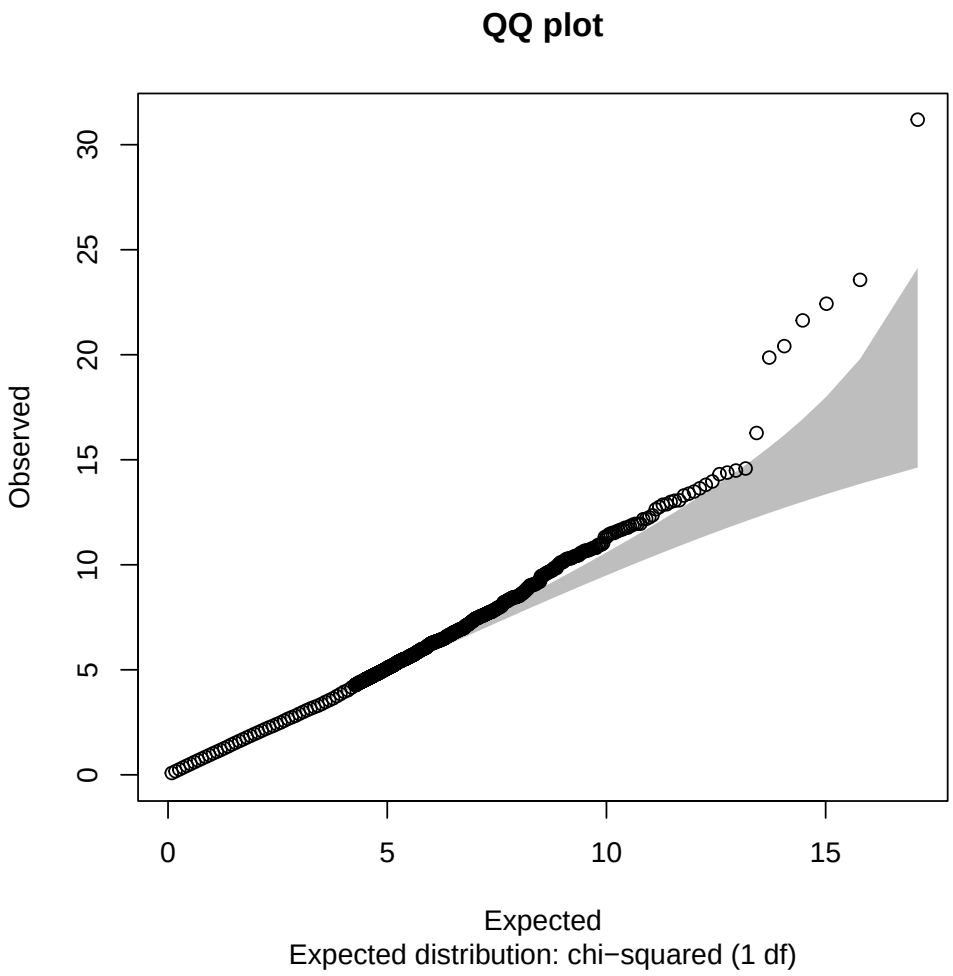
This over-dispersion of chi-squared values was built into our simulation. The data were constructed by re-sampling individuals from *two* groups of HapMap subjects, the CEU sample (of European origin) and the JPT+CHB sample (of Asian origin), The 55% of the cases were of European ancestry as compared with only 45% of the controls. We can deal with this by stratification of the tests, achieved by adding the `stratum` argument to the call to `single.snp.tests` (the following commands are as before)

```
> tests <- single.snp.tests(cc, stratum, data = subject.support,
+     snp.data = snps.10)
> tests <- tests[use, ]
> p1 <- pchisq(tests$chi2.1df, df = 1, lower.tail = FALSE)
> plot(hexbin(position, -log10(p1), xbin = 50))
```

[6]Note that this is not a simultaneous confidence region; the probability that the plot will stray outside the band at some point exceeds 95%.




```
> qq.chisq(tests$chi2.1df, df = 1)

           N      omitted       lambda
28184.000000     0.000000     1.006562
```



Most of the over-dispersion of test statistics has been removed (the residual is probably due to "cryptic relatedness" owing to the way in which the data were simulated).

Now let us find the names and positions of the most significant 10 SNPs. The first step is to compute an array which gives the positions in which the first, second, third etc. can be found

```
> ord <- order(p1)
> top10 <- ord[1:10]
> top10

 [1]   459 20174 20175 20173 20170 20171 20172 21134 26269  7981
```

We now list the 1 df p-values, the corresponding SNP names and their positions on the chromosome:

```
> names <- rownames(tests)
> p1[top10]

 [1] 2.336964e-08 1.206772e-06 2.179028e-06 3.296406e-06 6.248970e-06
 [6] 8.306560e-06 5.478332e-05 1.340763e-04 1.411405e-04 1.485893e-04

> names[top10]
```

```
 [1] "rs870041"   "rs10882596" "rs7088765"  "rs4918933"  "rs4918928"
 [6] "rs2025850"  "rs2274491"  "rs17668255" "rs7085895"  "rs11596495"
```

```
> position[top10]
```

```
 [1]   2075671  97190034  97191413  97189084  97179410  97185949  97186968
 [8] 101990691 127661165  33024457
```

The most associated SNPs lie within 3 small regions of the genome. To concentrate on the rightmost region (the most associated region on the left contains just one SNP), we'll first sort the names of the SNPs into position order along the chromosome and select those lying in the region approximately one mega-base either side of the second most associated SNP:

```
> posord <- order(position)
> position <- position[posord]
> names <- names[posord]
> local <- names[position > 9.6e+07 & position < 9.8e+07]
```

The variable `posord` contains the permutation necessary to sort SNPs into position order and `names` and `position` have now been reordered in this manner. the variable `local` contains the names of the SNPs in the selected 2 mega-base region. Now create a matrix containing just these SNPs, in position order, and compute the linkage disequilibrium (LD) between them:

```
> snps.2mb <- snps.10[, local]
> ld.2mb <- ld.snp(snps.2mb)
```

```
Information: The input contains 999 samples with 371 snps
... Done
```

A plot of the D' values across the region may be written to a file (in encapsulated postscript format) as follows:

```
plot(ld.2mb, file="ld2.eps")
```

This can be viewed (outside R) using a postscript viewer such as "gv" or "ggv". Alternatively it can be converted to a .pdf file and viewed in a pdf viewer such as "acroread". The associated SNPs fall in a region of tight LD towards the middle of the plot.

Next we shall estimate the size of the effect at the most associated SNPs for each region (rs870041, rs7088765, rs1916572). In the following commands, we extract this SNP from the matrix as a numerical variable (coded 0, 1, or 2) and then, using the `glm()` function, carry out a logistic regression of case–control status on this numerical coding of the SNP and upon stratum. The variable `stratum` must be included in the regression in order to allow for the different population structure of cases and controls. We first attach `subject.support` so that we can refer to `cc` and `stratum` variables directly:

```
> attach(subject.support)
> top <- snps.10[, "rs870041"]
> top <- as.numeric(top)
> glm(cc ~ top + stratum, family = "binomial")
```

```
Call:  glm(formula = cc ~ top + stratum, family = "binomial")

Coefficients:
   (Intercept)             top  stratumJPT+CHB
       -0.8953          0.5048         -0.2453

Degrees of Freedom: 998 Total (i.e. Null);  996 Residual
Null Deviance:            1385
Residual Deviance: 1345         AIC: 1351
```

The coefficient of top in this regression is estimated as 0.5048, equivalent to a relative equivalent to a relative risk of $\exp() = 1.657$. For the other top SNPs we have:

```
> top2 <- snps.10[, "rs7088765"]
> top2 <- as.numeric(top2)
> glm(cc ~ top2 + stratum, family = "binomial")

Call:  glm(formula = cc ~ top2 + stratum, family = "binomial")

Coefficients:
   (Intercept)            top2  stratumJPT+CHB
        1.0238         -0.4097         -0.4978

Degrees of Freedom: 998 Total (i.e. Null);  996 Residual
Null Deviance:            1385
Residual Deviance: 1358         AIC: 1364

> top3 <- snps.10[, "rs1916572"]
> top3 <- as(top3, "numeric")
> glm(cc ~ top3 + stratum, family = binomial)

Call:  glm(formula = cc ~ top3 + stratum, family = binomial)

Coefficients:
   (Intercept)            top3  stratumJPT+CHB
       -0.4752          0.3783         -0.2189

Degrees of Freedom: 986 Total (i.e. Null);  984 Residual
  (12 observations deleted due to missingness)
Null Deviance:            1368
Residual Deviance: 1350         AIC: 1356
```

So the relative risks are, respectively, $\exp(-0.4097) = 0.664$ and $\exp(0.3783) = 1.460$.

Finally you might like to repeat the analysis above using the 2 df tests. The conclusion would have been much the same. A word of caution however; with real data the 2 df test is less robust against artifacts due to genotyping error. On the other hand, it is much more powerful against recessive or near-recessive variants.

Advanced topics: multi-locus tests

There are two other functions for carrying out association tests (`snp.lhs.tests()` and `snp.rhs.tests()`) in the package. These are somewhat slower, but much more flexible. For example, the former function allows one to test for differences in allele frequencies between more than two groups. An important use of the latter function is to carry out tests using *groups* of SNPs rather than single SNPs. We shall explore this use in the final part of the exercise.

A prerequisite to multi-locus analyses is to decide on how SNPs should be grouped in order to "tag" the genome rather more completely than by use of single markers. The `snpMatrix` package will eventually contain tools to compute such groups, for example, by using HapMap data. The function `ld.snp()`, which we encountered earlier, will be an essential tool in this process. However this work is not complete and, for now, we demonstrate the testing tool by grouping the 27,828 SNPs we have decided to use into 20kB blocks. The following commands compute such a grouping, tabulate the block size, and remove empty blocks:

```
> blocks <- split(posord, cut(position, seq(1e+05, 135300000, 20000)))
> bsize <- sapply(blocks, length)
> table(bsize)

bsize
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
803 732 895 869 801 665 581 417 316 192 170 102  72  41  43  20  13   9   5   5
 20  21  22  24
  1   6   1   1

> blocks <- blocks[bsize > 0]
```

You can check that this has worked by listing the column positions of the first 20 SNPs together with the those contained in the first five blocks

```
> posord[1:20]

 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

> blocks[1:5]

$`(1e+05,1.2e+05]`
[1] 1 2 3

$`(1.2e+05,1.4e+05]`
[1] 4

$`(1.4e+05,1.6e+05]`
[1]  5  6  7  8  9 10

$`(1.6e+05,1.8e+05]`
[1] 11 12 13 14

$`(1.8e+05,2e+05]`
[1] 15 16 17 18
```

Note that these positions refer to the reduced set of SNPs after application of the filter on MAF and HWE. Therefore, before proceeding further we create a new matrix of SNP genotypes containing only these 27,828:

```
> snps.use <- snps.10[, use]
> remove(snps.10)
```

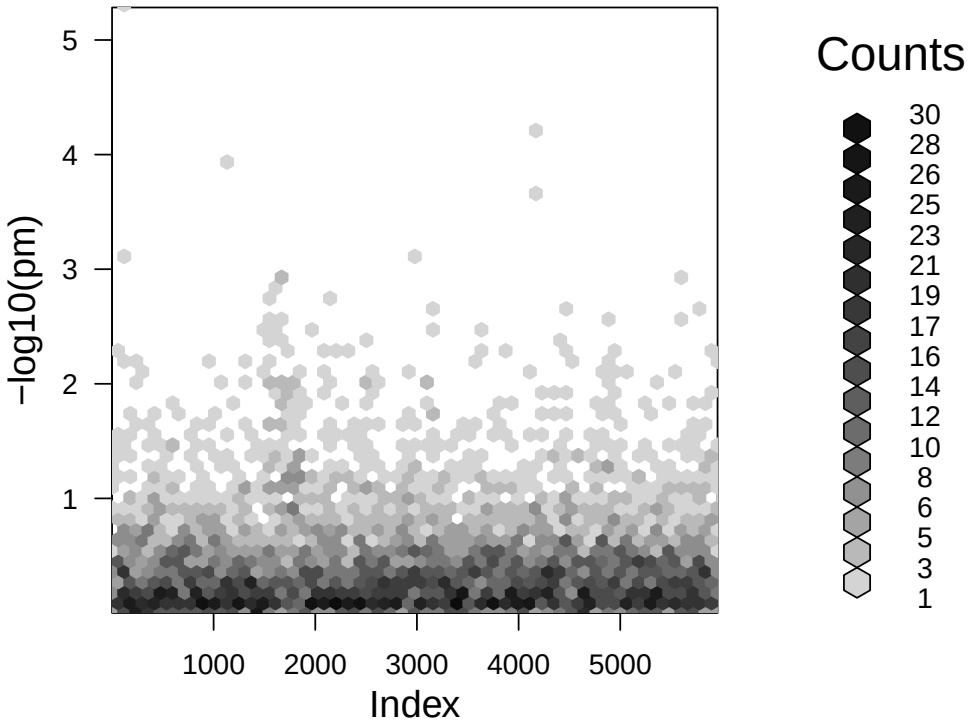
The command to carry out the tests on these groups, controlling for the known population structure differences is

```
> mtests <- snp.rhs.tests(cc ~ stratum, family = "binomial",
+      data = subject.support, snp.data = snps.use, tests = blocks)
> summary(mtests)
```

```
  Chi.squared              Df          Df.residual
 Min.   : 0.000004   Min.   : 1.000   Min.   :771.0
 1st Qu.: 1.273069   1st Qu.: 2.000   1st Qu.:935.0
 Median : 3.162559   Median : 4.000   Median :957.0
 Mean   : 4.161161   Mean   : 4.148   Mean   :951.1
 3rd Qu.: 5.994639   3rd Qu.: 6.000   3rd Qu.:974.0
 Max.   :32.290273   Max.   :24.000   Max.   :995.0
```

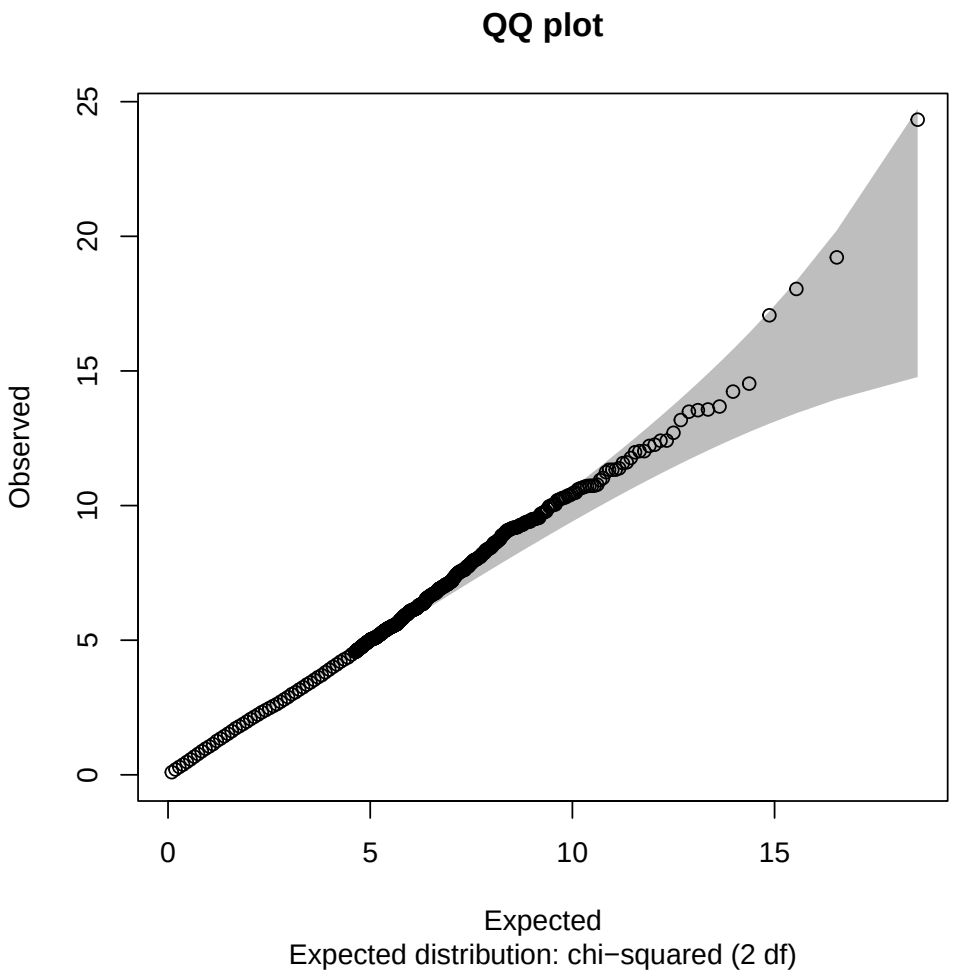
The first argument, together with the second, specifies the model which corresponds to the null hypothesis. In this case we have allowed for the variation in ethnic origin (`stratum`) between cases and controls. We complete the analysis by calculating the p–values and plotting minus their logs (base 10):

```
> pm <- pchisq(mtests$Chi.squared, mtests$Df, lower.tail = FALSE)
> plot(hexbin(-log10(pm), xbin = 50))
```



The same associated region is picked out, albeit with a rather larger p-value; in this case the multiple df test cannot be powerful as the 1 df test since the simulation ensured that the "causal" locus was actually one of the SNPs typed on the Affymetrix platform. QQ plots are somewhat more difficult since the tests are on differing degrees of freedom. This difficulty is neatly circumvented by noting that $-2\log p$ is, under the null hypothesis, distributed as chi-squared on 2 df:

```
> qq.chisq(-2 * log(pm), df = 2)

          N     omitted      lambda
5957.000000    0.000000    1.037329
```


QQ plot
Observed
0
5
10
15
20
25
0
5
10
15
Expected
Expected distribution: chi−squared (2 df)