snpStats vignette
Example of genome-wide association testing

David Clayton and Chris Wallace

October 24, 2025

The `snpMatrix` and `snpStats` packages

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.

`snpStats` arose out of the need to store, and analyse, SNP genotype data in which subjects cannot be assigned to the three possible genotypes with certainty. This necessitated a change in the way in which data are stored internally, although `snpStats` can still handle conventionally called genotype data stored in the original `snpMatrix` storage mode. `snpStats` currently lacks some facilities which were present in `snpMatrix` (although, hopefully, the important gaps will soon be filled) but it also includes several important new facilities. This vignette currently exploits none of the new facilities; these are mainly used in the vignette which deals with imputation and meta-analysis.

For population-based studies, both quantitative and qualitative phenotypes may be analysed but, at present, rather more limited facilities are available for family–based studies. Flexible 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. The original `snpMatrix` package was described by Clayton and Leung (2007) *Human Heredity*, **64**: 45–51. Since this publication many new facilities have been introduced; some of these are explored in further vignettes.

Getting started

We shall start by loading the the packages and the data to be used in the first part of this exercise, which concerns a population–based case–control study:

```
> require(snpStats)
> require(hexbin)
> data(for.exercise)
```

In addition to the `snpStats` 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 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 "`SnpMatrix`" containing a matrix of SNP genotype calls. Rows of the matrix correspond to subjects and columns correspond to SNPs:

   ```
   > show(snps.10)

   A SnpMatrix 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.   :1.02e+05   A:14019   C: 2349
   1st Qu.:10   1st Qu.:2.90e+07   C:12166   G:12254
   Median :10   Median :6.74e+07   G: 2316   T:13898
   Mean   :10   Mean   :6.69e+07
   3rd Qu.:10   3rd Qu.:1.02e+08
   Max.   :10   Max.   :1.35e+08
   ```

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

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

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

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
```

   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].

A short summary of the contents of `snps.10` is provided by the `summary` function. This operation actually produces two "summaries of summaries". First, summary statistics are calculated for each row (sample), and their results summarised. Then summary statistics are calculated for each column (SNP) and their results summarised.

```
> summary(snps.10)
```

```
$rows
   Call.rate     Certain.calls Heterozygosity
 Min.   :0.988   Min.   :1     Min.   :0.000
 1st Qu.:0.990   1st Qu.:1     1st Qu.:0.299
 Median :0.990   Median :1     Median :0.308
 Mean   :0.990   Mean   :1     Mean   :0.307
 3rd Qu.:0.990   3rd Qu.:1     3rd Qu.:0.316
 Max.   :0.992   Max.   :1     Max.   :0.339

$cols
     Calls        Call.rate     Certain.calls      RAF              MAF
 Min.   : 975   Min.   :0.975   Min.   :1     Min.   :0.000   Min.   :0.000
 1st Qu.: 988   1st Qu.:0.988   1st Qu.:1     1st Qu.:0.230   1st Qu.:0.126
 Median : 990   Median :0.990   Median :1     Median :0.503   Median :0.232
```

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

```
Mean   : 990   Mean   :0.990   Mean   :1     Mean   :0.500   Mean   :0.242
3rd Qu.: 992   3rd Qu.:0.992   3rd Qu.:1     3rd Qu.:0.767   3rd Qu.:0.358
Max.   :1000   Max.   :1.000   Max.   :1     Max.   :1.000   Max.   :0.500

     P.AA             P.AB            P.BB            z.HWE
Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :-21.973
1st Qu.:0.0656   1st Qu.:0.208   1st Qu.:0.0646   1st Qu.: -2.850
Median :0.2688   Median :0.320   Median :0.2749   Median : -1.191
Mean   :0.3462   Mean   :0.307   Mean   :0.3465   Mean   : -1.861
3rd Qu.:0.6059   3rd Qu.:0.422   3rd Qu.:0.6036   3rd Qu.: -0.101
Max.   :1.0000   Max.   :0.550   Max.   :1.0000   Max.   :  3.708
                                                  NA's   :4
```

The row-wise and column-wise summaries are calculated with the functions `row.summary` and `col.summary`. For example, to calculate summary statistics for each SNP (column):

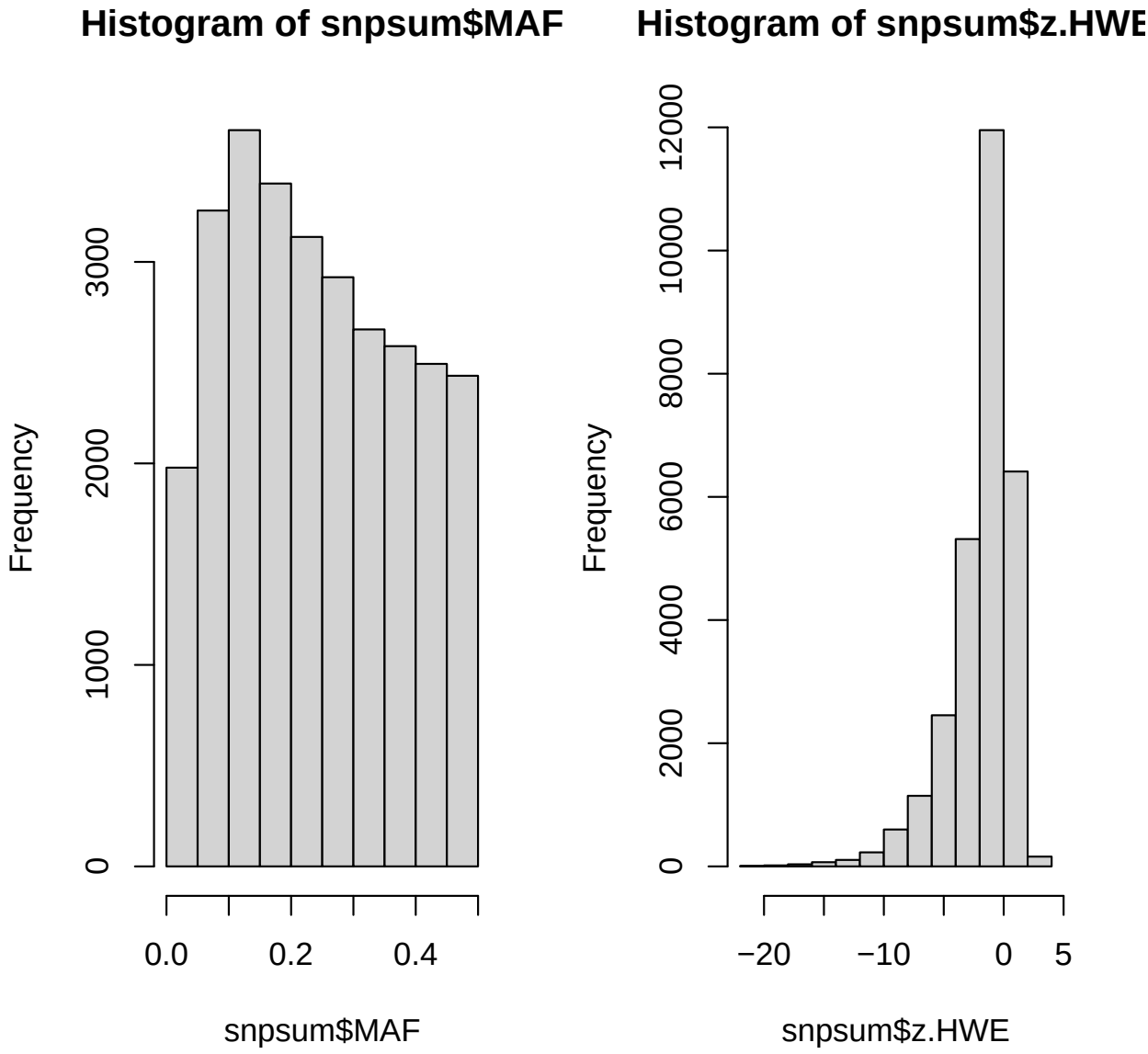
```
> snpsum <- col.summary(snps.10)
> summary(snpsum)

    Calls        Call.rate     Certain.calls      RAF             MAF
Min.   : 975   Min.   :0.975   Min.   :1     Min.   :0.000   Min.   :0.000
1st Qu.: 988   1st Qu.:0.988   1st Qu.:1     1st Qu.:0.230   1st Qu.:0.126
Median : 990   Median :0.990   Median :1     Median :0.503   Median :0.232
Mean   : 990   Mean   :0.990   Mean   :1     Mean   :0.500   Mean   :0.242
3rd Qu.: 992   3rd Qu.:0.992   3rd Qu.:1     3rd Qu.:0.767   3rd Qu.:0.358
Max.   :1000   Max.   :1.000   Max.   :1     Max.   :1.000   Max.   :0.500

     P.AA             P.AB            P.BB            z.HWE
Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :-21.973
1st Qu.:0.0656   1st Qu.:0.208   1st Qu.:0.0646   1st Qu.: -2.850
Median :0.2688   Median :0.320   Median :0.2749   Median : -1.191
Mean   :0.3462   Mean   :0.307   Mean   :0.3465   Mean   : -1.861
3rd Qu.:0.6059   3rd Qu.:0.422   3rd Qu.:0.6036   3rd Qu.: -0.101
Max.   :1.0000   Max.   :0.550   Max.   :1.0000   Max.   :  3.708
                                                  NA's   :4
```

The second command duplicates the latter part of the result of `summary(snps.10)`, and the contents of `snpsum` are fairly self-explanatory. We could look at a couple of summary statistics in more detail:

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



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.

The function `row.summary` is useful for detecting samples that have genotyped poorly. This calculates call rate and mean heterozygosity across all SNPs for each subject in turn:

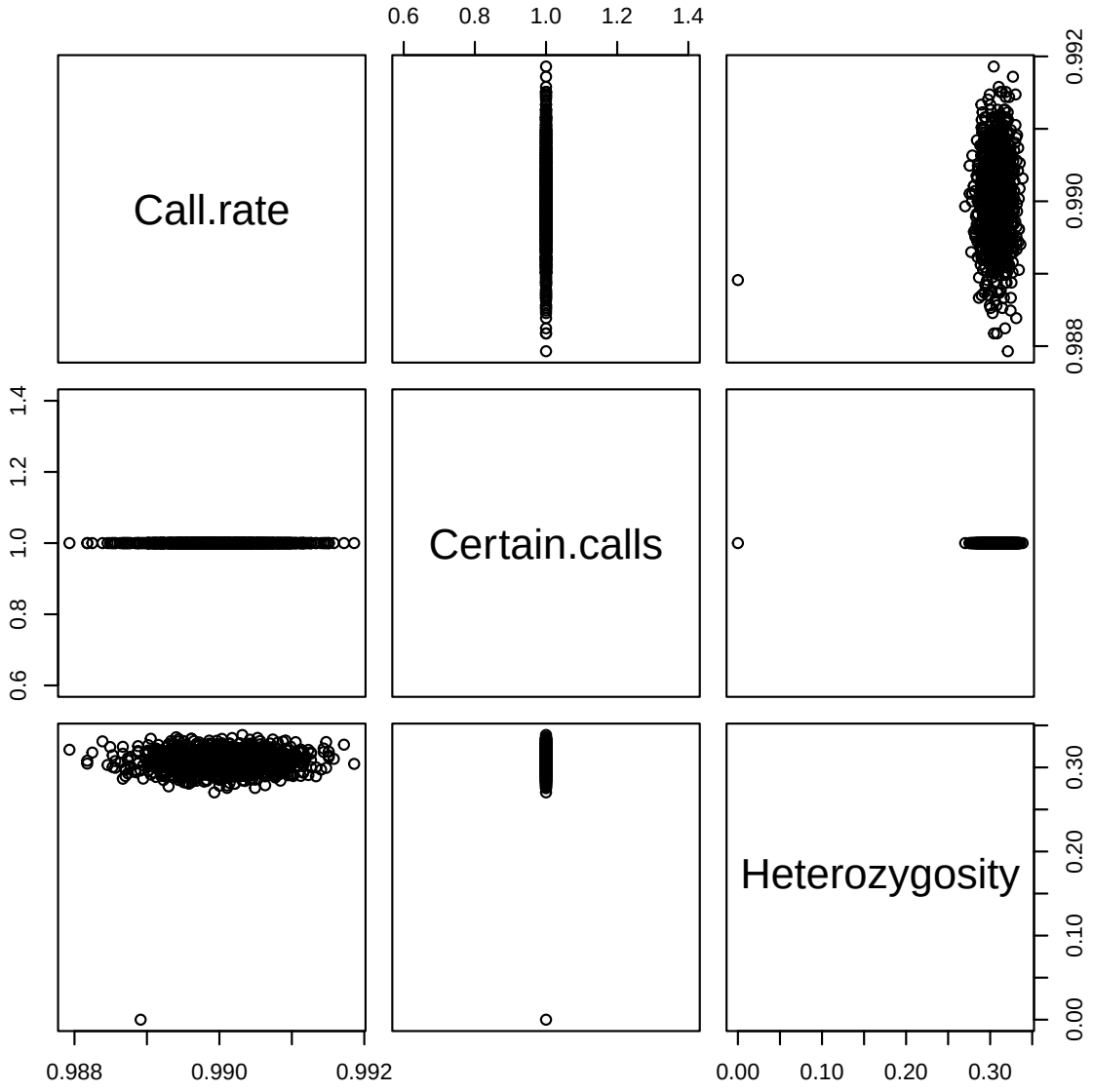
```
> sample.qc <- row.summary(snps.10)
> summary(sample.qc)

   Call.rate     Certain.calls Heterozygosity
```

```
 Min.   :0.988   Min.   :1     Min.   :0.000
 1st Qu.:0.990   1st Qu.:1     1st Qu.:0.299
 Median :0.990   Median :1     Median :0.308
 Mean   :0.990   Mean   :1     Mean   :0.307
 3rd Qu.:0.990   3rd Qu.:1     3rd Qu.:0.316
 Max.   :0.992   Max.   :1     Max.   :0.339
```

(note that the last command yields the same as the first part of `summary(snps.10)`). The plot of heterozygosity against call rate is useful in detecting poor quality samples:

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



There is one clear outlier.

The analysis

We'll start by removing the 'outlying' sample above (the sample 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
```

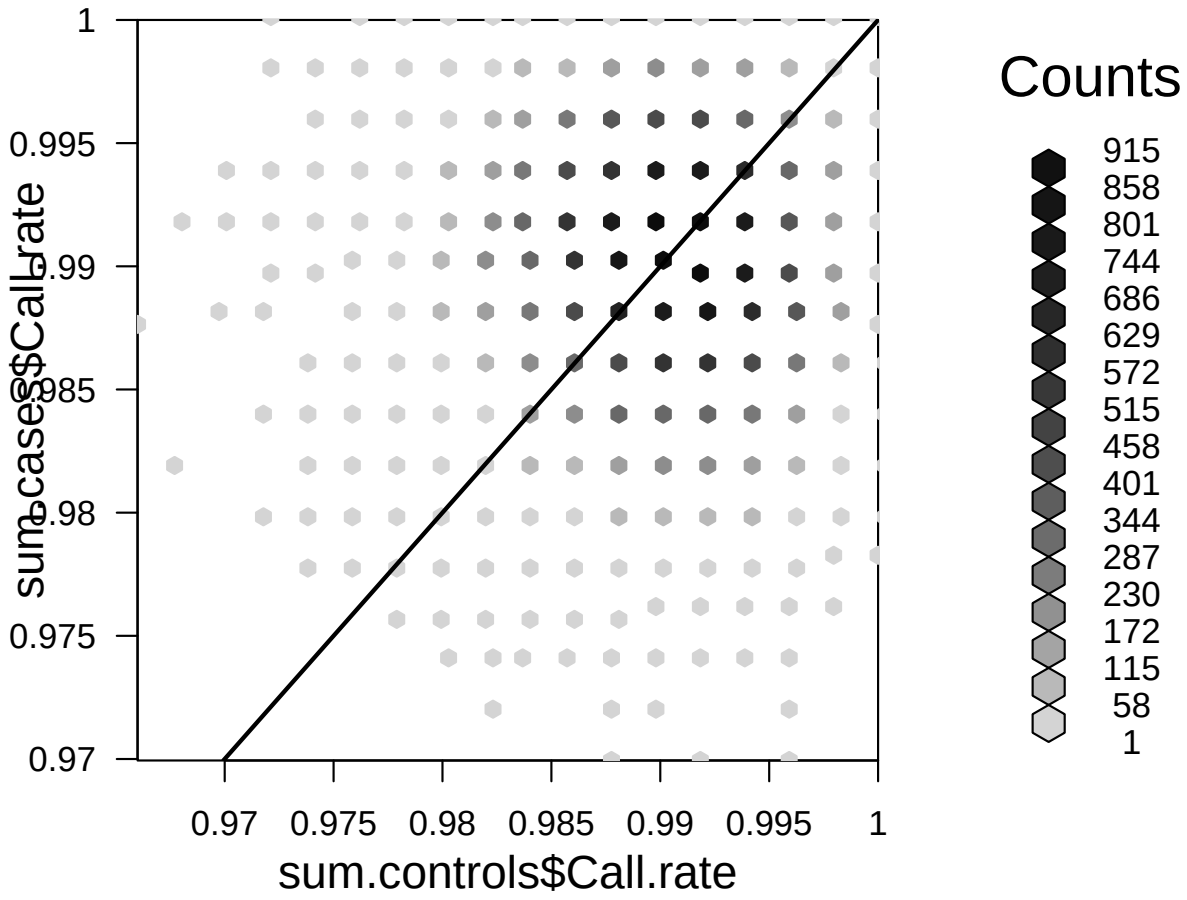
Now we recompute the genotype column summaries separately for cases and controls:

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

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

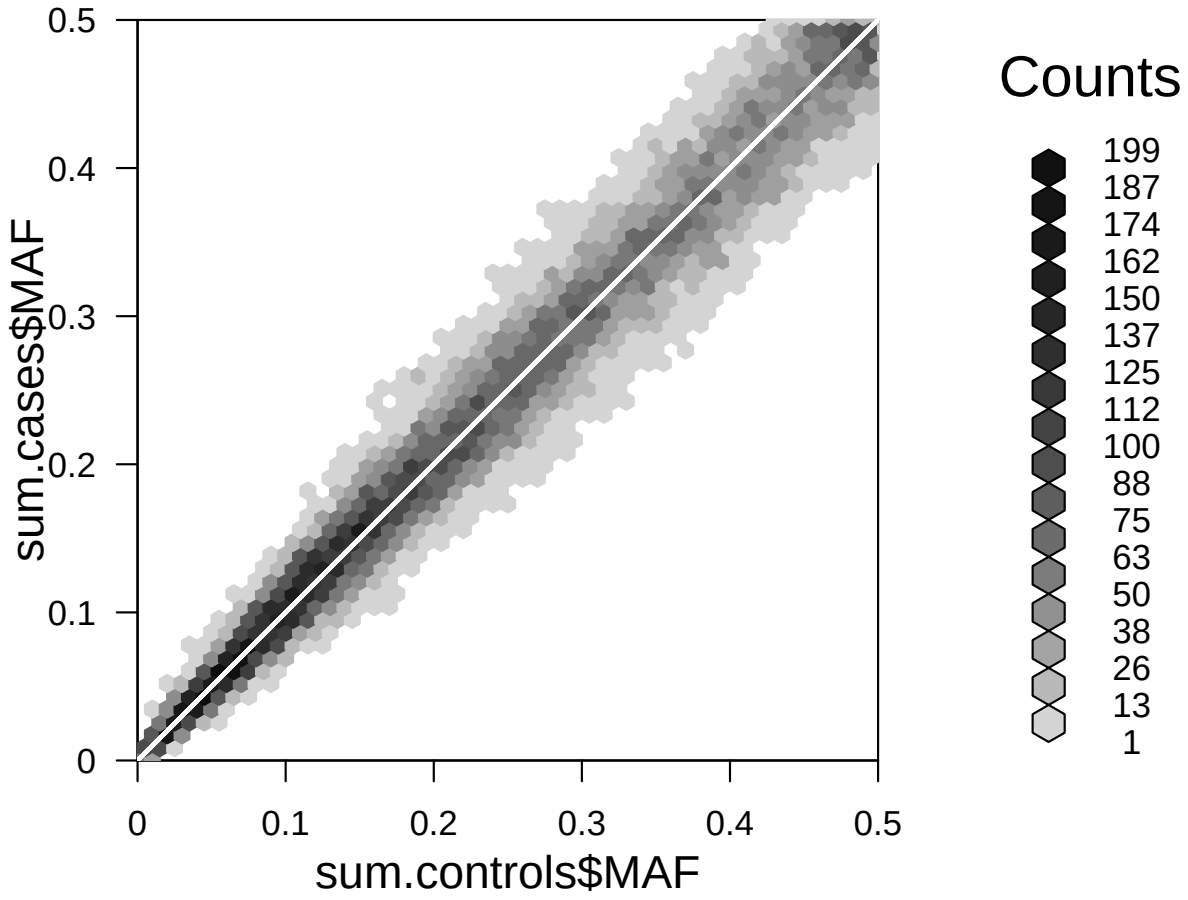
```
> hb <- hexbin(sum.controls$Call.rate, sum.cases$Call.rate, xbin=50)
> sp <- plot(hb)
> hexVP.abline(sp$plot.vp, 0, 1, col="black")
```

[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.



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, 0, 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)
```

```
       N        Chi.squared.1.df Chi.squared.2.df     P.1df
 Min.   :974   Min.   : 0.000   Min.   : 0.00    Min.   :0.000
 1st Qu.:987   1st Qu.: 0.172   1st Qu.: 0.79    1st Qu.:0.141
 Median :989   Median : 0.773   Median : 1.86    Median :0.379
 Mean   :989   Mean   : 1.561   Mean   : 2.60    Mean   :0.419
 3rd Qu.:991   3rd Qu.: 2.167   3rd Qu.: 3.67    3rd Qu.:0.678
 Max.   :999   Max.   :34.022   Max.   :37.25    Max.   :1.000
               NA's   :4        NA's   :830      NA's   :4
     P.2df
 Min.   :0.000
 1st Qu.:0.160
 Median :0.395
 Mean   :0.428
 3rd Qu.:0.674
 Max.   :1.000
 NA's   :830
```

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)
```

[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.

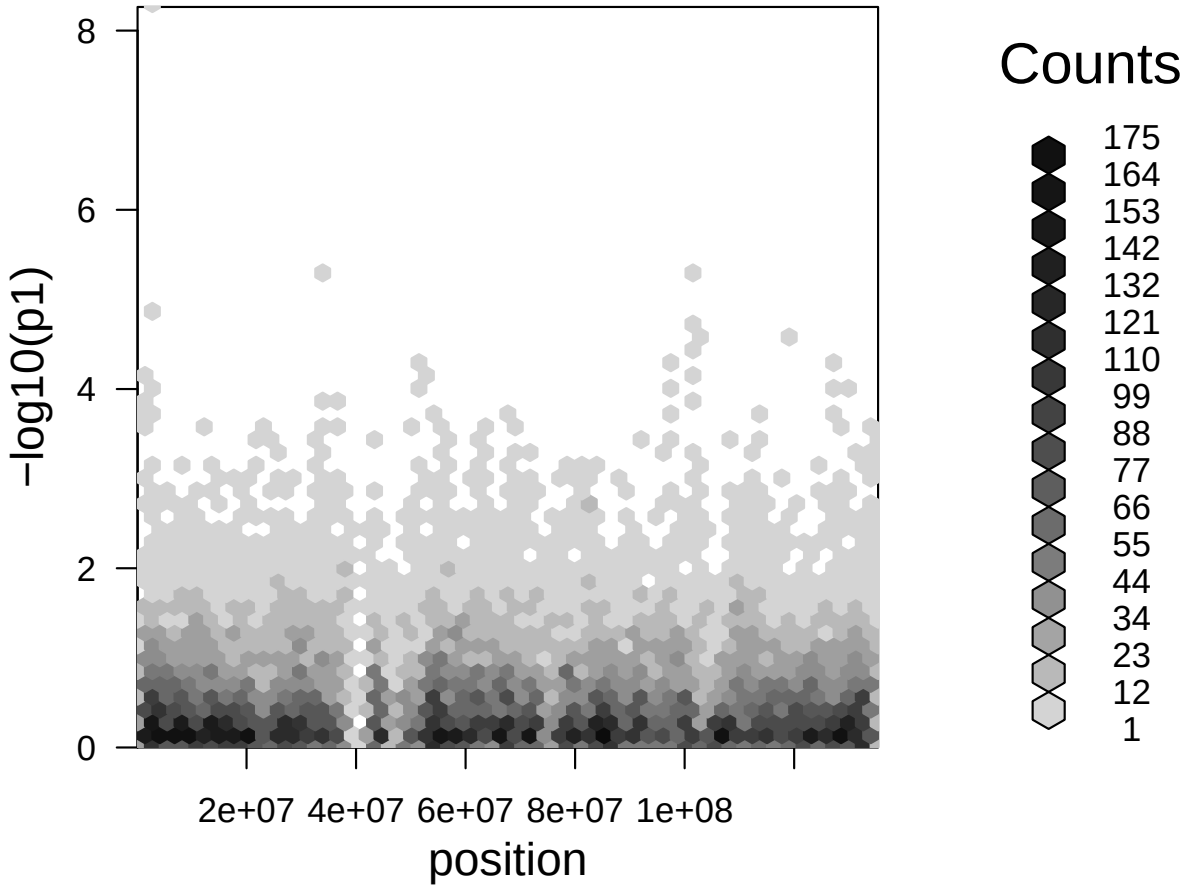
```
[1] 28184
```

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

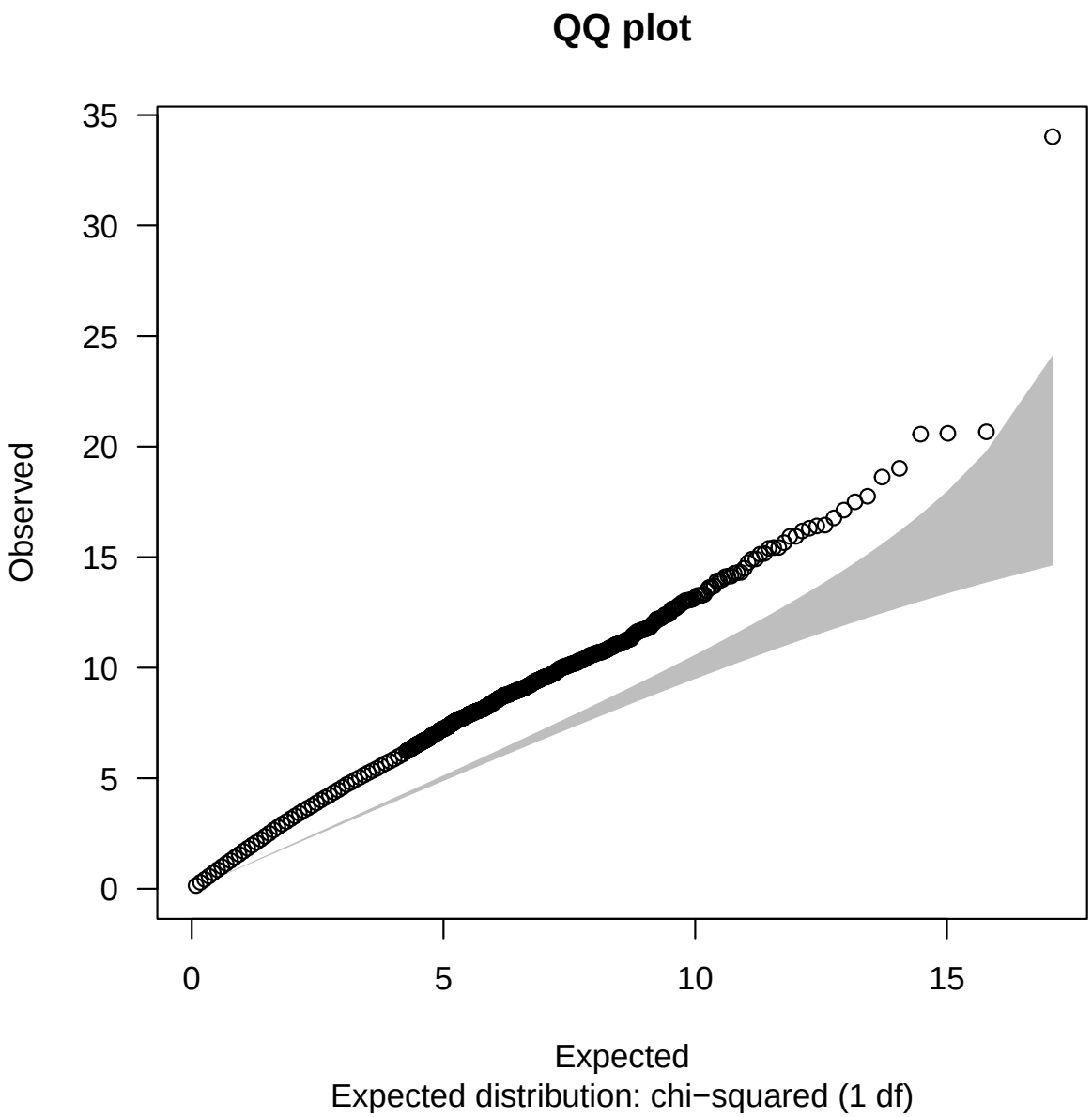
```
> p1 <- p.value(tests, df=1)
> plot(hexbin(position, -log10(p1), xbin=50))
```



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

```
> chi2 <- chi.squared(tests, df=1)
> qq.chisq(chi2,  df = 1)

        N   omitted    lambda
28184.000     0.000     1.677
```



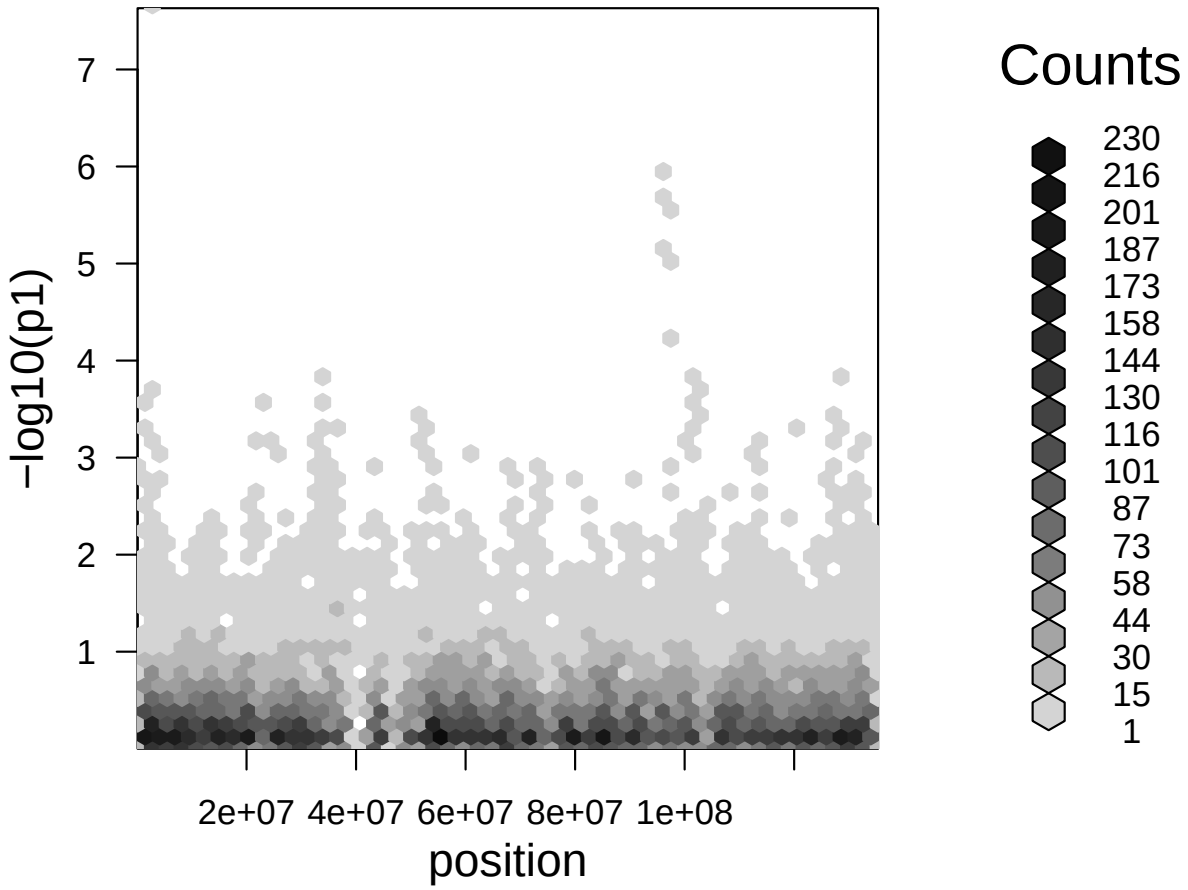
The three numbers returned by this command are the number of tests considered, the number of outliers falling beyond the plot boundary, and the slope of a line fitted to the smallest 90% of values (*i.e.* the multiple by which the chi-squared test statistics are over-dispersed). The "concentration band" 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 sam-

[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%.

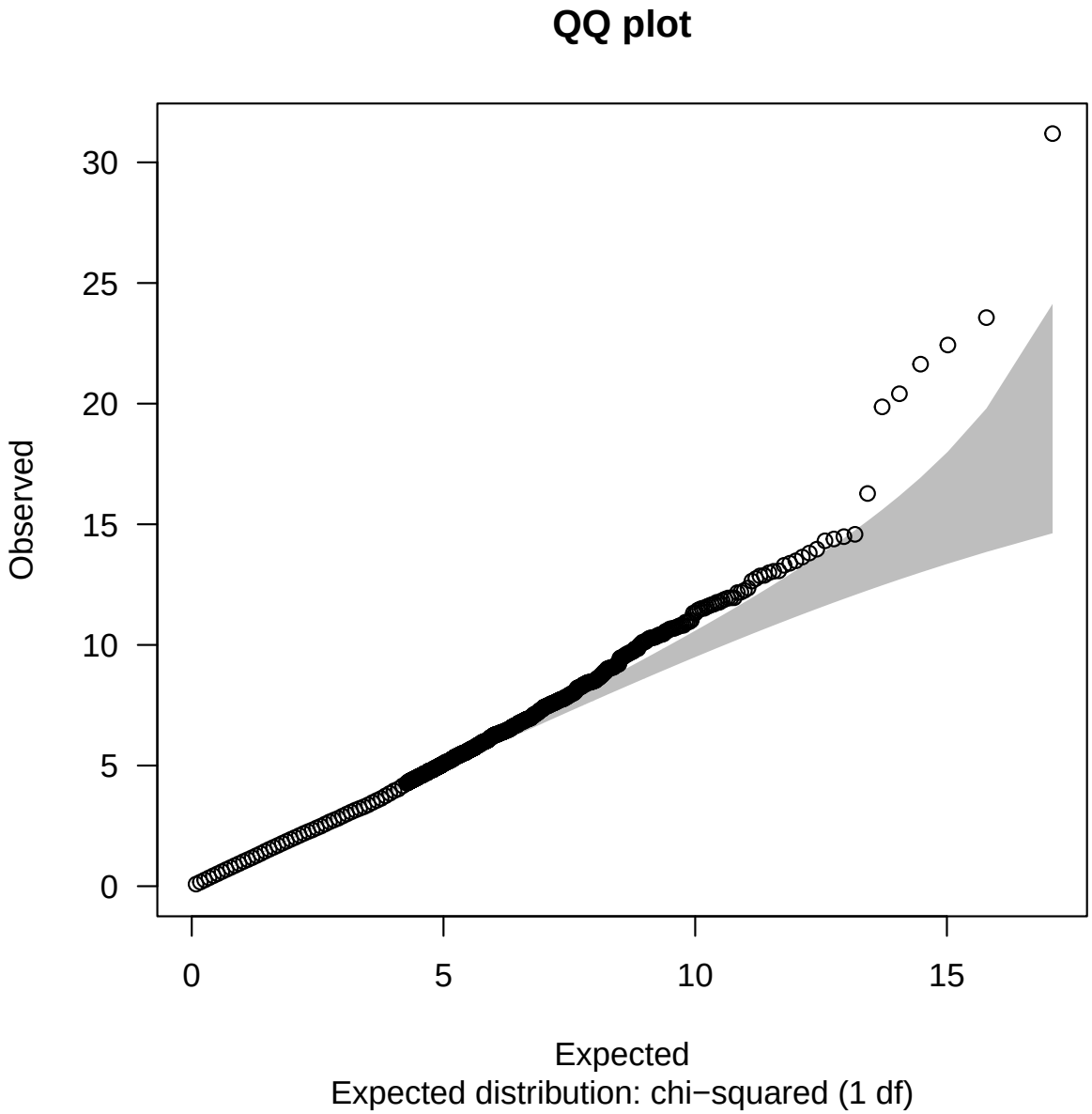
ple (of European origin) and the JPT+CHB sample (of Asian origin). The 55% of the cases were of European ancestry as compared with only 45% of the controls. We can deal with this by stratification of the tests, achieved by adding the `stratum` argument to the call to `single.snp.tests` (the remaining commands are as before)

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



```
> chi2 <- chi.squared(tests, df=1)
> qq.chisq(chi2, df = 1)

        N   omitted    lambda
28184.000     0.000     1.007
```



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 <- tests@snp.names
> p1[top10]
```

```
  rs870041 rs10882596  rs7088765  rs4918933  rs4918928  rs2025850  rs2274491
 2.337e-08  1.207e-06  2.179e-06  3.296e-06  6.249e-06  8.307e-06  5.478e-05
rs17668255  rs7085895 rs11596495
 1.341e-04  1.411e-04  1.486e-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 two 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` now 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.

Next we shall estimate the size of the effect at the most associated SNPs for each region (rs870041, rs10882596). In the following commands, we extract each 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 make copies of the `cc` and `stratum` variables in `subject.support` in the current working environment (where the other variables reside):

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

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

Coefficients:
   (Intercept)             top  stratumJPT+CHB
        -0.405           0.510          -0.230

Degrees of Freedom: 988 Total (i.e. Null);  986 Residual
  (10 observations deleted due to missingness)
Null Deviance:            1370
Residual Deviance: 1330         AIC: 1340
```

The coefficient of `top` in this regression is estimated as 0.5100, equivalent to a relative risk of $\exp(.5100) = 1.665$. For the other top SNP we have:

```
> top2 <- as(snps.10[, "rs10882596"], "numeric")
> fit <- glm(cc ~ top2 + stratum, family = "binomial")
> summary(fit)

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

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)      -0.245      0.123   -2.00  0.04556 *
top2              0.458      0.095    4.82  1.4e-06 ***
stratumJPT+CHB   -0.512      0.136   -3.76  0.00017 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1373.8  on 990  degrees of freedom
Residual deviance: 1343.9  on 988  degrees of freedom
  (8 observations deleted due to missingness)
AIC: 1350

Number of Fisher Scoring iterations: 4
```

This relative risk is $\exp(0.4575) = 1.580$. Both estimates are close to the values used to simulate the data.

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.

The `snpStats` package includes its own functions to fit generalized linear models. These are much faster than `glm`, although not yet as flexible. They allow for a each of series of SNPs to be entered into a GLM, either on the left hand side (*i.e.* as the dependent variable) or on the right-hand side (as a predictor variable). In the latter case seveal SNPs can be entered in each model fit. For example, to fit the same GLM as before, in which each SNP is entered in turn on the right-hand side of a logistic regression equation, for each of the SNPs in the 2 megabase "local" region:

```
> localest <- snp.rhs.estimates(cc~stratum, family="binomial", sets=local,
+                               data=subject.support, snp.data=snps.10)
```

This function call has computed 371 GLM fits! The parameter estimates for the first five, and for the second best SNP analyzed above (rs10882596) are shown by

```
> localest[1:5]
```

Model	Y-variable	Parameter	Estimate	S.E.	z-value
rs9787457	cc	rs9787457	0.14176	0.1016	1.395
rs7919320	cc	rs7919320	0.15527	0.10145	1.530
rs4918184	cc	rs4918184	0.075481	0.10391	0.726
rs11187837	cc	rs11187837	-0.031994	0.12002	-0.267
rs7084339	cc	rs7084339	0.10027	0.10235	0.980

```
> localest["rs10882596"]
```

Model	Y-variable	Parameter	Estimate	S.E.	z-value
rs10882596	cc	rs10882596	0.4575	0.094955	4.818

The parameter estimate for rs1088259 and its standard error agree closely with the values obtained earlier, using the `glm` function.

The GLM code within `snpStats` allows a further speed-up which is not available in the standard `glm` function. If a variable is to be included in the model as a "factor" taking many levels then a more efficient algorithm can be invoked by using the `strata` function in the model formula. For example, the following command fits the same model for all the 28184 SNPs we have decided to use in these analyses:

```
> allest <- snp.rhs.estimates(cc~strata(stratum), family="binomial", sets=use,
+                                 data=subject.support, snp.data=snps.10)
> length(allest)

[1] 28184
```

As expected, the parameter estimates and standard errors are unchanged, for example:

```
> allest["rs10882596"]

     Model  Y-variable   Parameter    Estimate        S.E.     z-value
-------------------------------------------------------------------------
rs10882596          cc  rs10882596      0.4575    0.094955       4.818
-------------------------------------------------------------------------
```

Note that `strata()` can only be used once in a model formula.

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. Hopefully, 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 28184 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(100000, 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)
```

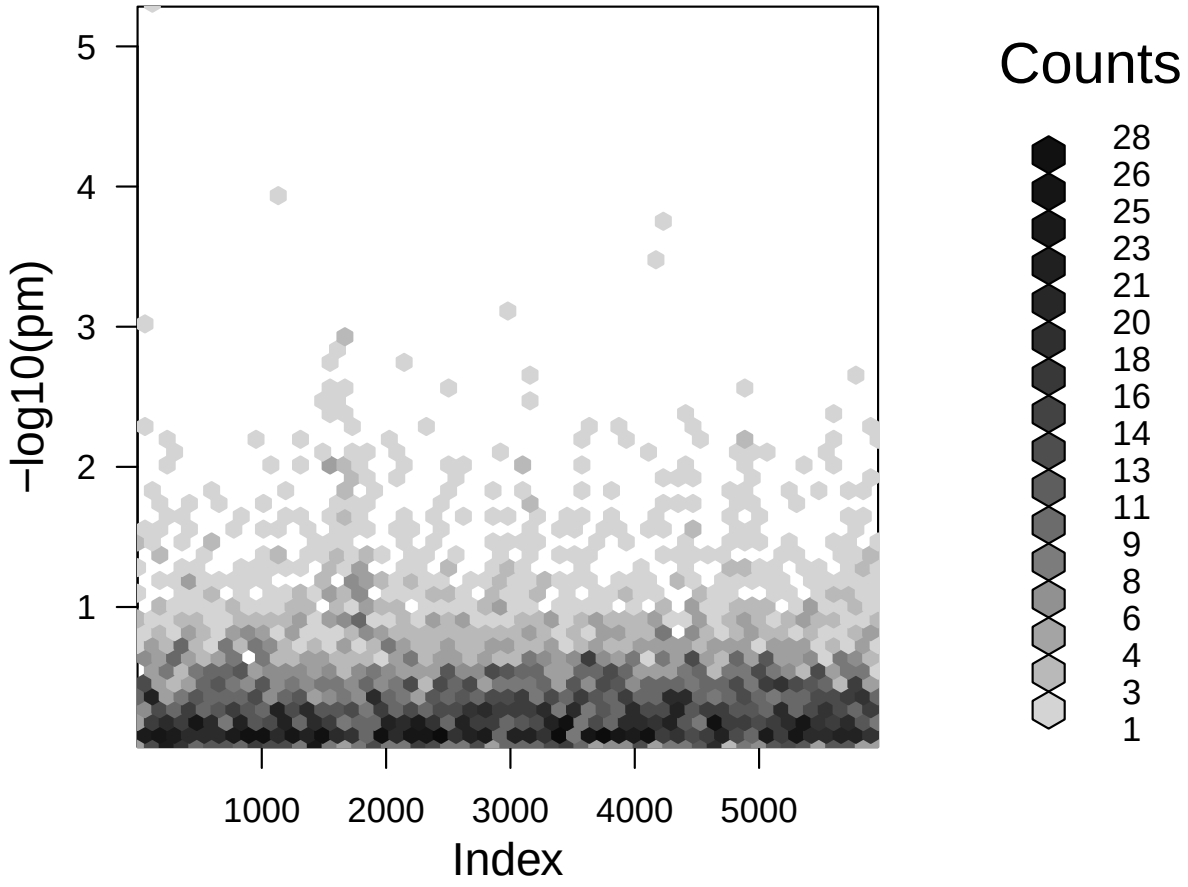
The command to carry out the tests on these groups, controlling for the known population structure differences is

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

  Chi.squared          Df           p.value
 Min.   : 0.00   Min.   : 1.00   Min.   :5.20e-06
 1st Qu.: 1.44   1st Qu.: 2.00   1st Qu.:2.58e-01
 Median : 3.48   Median : 4.00   Median :4.88e-01
 Mean   : 4.52   Mean   : 4.51   Mean   :4.95e-01
 3rd Qu.: 6.53   3rd Qu.: 6.00   3rd Qu.:7.40e-01
 Max.   :32.29   Max.   :24.00   Max.   :1.00e+00
```

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 extracting the p–values and plotting minus their logs (base 10):

```
> pm <- p.value(mtests)
> 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, under the null hypothesis, $-2 \log p$ is distributed as chi-squared on 2 df:

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

       N  omitted   lambda
5957.000    0.000    1.056
```