SNPassoc: an R package to perform whole genome association studies

Juan R González, Lluís Armengol, Xavier Solé,
Elisabet Guinó, Josep M Mercader, Xavier Estivill, Víctor Moreno

March 2, 2007

Contents

The SNPassoc package contains facilities for data manipulation, tools for exploratory data analysis, convenient graphical facilities, and tools for assessing genetic association for

both quantitative and categorial (case-control) traits in whole genome approaches. Genome-based studies are normally analyzed using a multistage approach. In the first step researchers are interested in assessing association between the outcome and thousands of SNPs (Section 2). In a second and possibly third step, medium/large scale studies are performed in which only a few hundred of SNPs, those with a putative association found in the first step, are genotyped (Section 3). `SNPassoc` is specially designed for analyzing this kind of designs. In addition, a convenience-based approach (select variants on the basis of logistical considerations such as the ease and cost of genotyping) can also be analyzed using `SNPassoc`. Different genetic models are also implemented in the package. Analysis of multiple SNPs can be analyzed using either haplotype or gene-gene interaction approaches (Section 4). Statistical methods used in the functions are described in Section 5. Lastly, some computational aspects are described in Section 6.

1 Data manipulation and descriptive analysis

1.1 The class `snp`

Let's assume that the data set we are analyzing looks like this

```
#Let's load library SNPassoc
>library(SNPassoc)

#get the data example:
#both data.frames SNPs and SNPs.info.pos are loaded typing data(SNPs)
>data(SNPs)

#look at the data (only first four SNPs)
> SNPs[1:10,1:9]
   id casco    sex blood.pre  protein snp10001 snp10002 snp10003 snp10004
1   1     1 Female      13.7 75640.52       TT       CC       GG       GG
2   2     1 Female      12.7 28688.22       TT       AC       GG       GG
3   3     1 Female      12.9 17279.59       TT       CC       GG       GG
4   4     1   Male      14.6 27253.99       CT       CC       GG       GG
5   5     1 Female      13.4 38066.57       TT       AC       GG       GG
6   6     1 Female      11.3  9872.46       TT       CC       GG       GG
7   7     1 Female      11.9 11132.90       TT       AC       GG       GG
8   8     1   Male      12.4 29973.43       TT       AC       GG       GG
9   9     1   Male      14.5 31114.29       CT       CC       GG       GG
10 10     1 Female      12.2 41768.55       TT       AC       GG       GG

... etc
```

The function `snp` has been designed for dealing with SNP variables. Here `SNPassoc` uses the object-oriented features of `R` ("classes and methods") to make it easy to manipulate, analyze, and plot data sets. Notice that the two alleles in a genotype are normally separated by a given character. However, users may employ different formats just by changing the argument called `sep`. In our example `sep=""` since there is no character between the two alleles. To homogenize the results we decided to separate both alleles by "/" when an object of class `snp` is created.

```
> mySNP<-snp(SNPs$snp10001,sep="")
> mySNP
  [1] T/T T/T T/T C/T T/T T/T T/T T/T C/T T/T C/T C/C C/T T/T T/T T/T C/T T/T
 [19] T/T T/T T/T C/C C/T T/T T/T C/T T/T T/T T/T C/C T/T T/T C/T T/T C/T C/T
 [37] C/C C/T C/T T/T T/T T/T T/T C/T T/T C/C C/T C/T C/T T/T T/T C/T C/T T/T
```

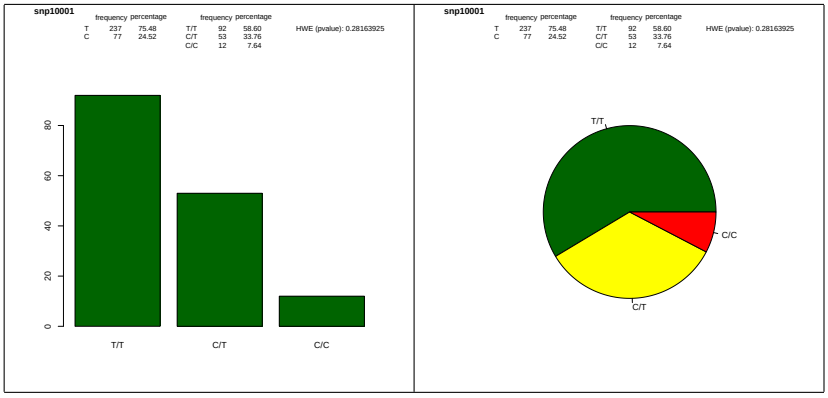

Figure 1: Summary for a given SNP using `barplot` (left figure) or `pie` (right figure) R graphics.

```
 [55] T/T T/T T/T T/T C/T T/T T/T T/T T/T T/T T/T T/T T/T C/C T/T T/T C/T C/T T/T
 [73] T/T C/T T/T T/T T/T T/T T/T C/T C/T T/T T/T T/T C/C T/T T/T C/C C/C C/T
 [91] T/T T/T C/T T/T T/T T/T T/T T/T C/T T/T C/T C/T C/T T/T T/T C/C C/T T/T
[109] T/T C/T T/T C/T C/T T/T C/T C/T T/T T/T C/T C/T C/T T/T C/C T/T T/T T/T
[127] T/T T/T C/T T/T C/T C/C T/T C/T C/T T/T C/T T/T T/T C/T C/T C/T T/T T/T
[145] T/T T/T C/T T/T T/T T/T T/T C/T T/T C/T C/T C/T C/T
Levels: T/T C/T C/C
```

An object of class 'snp' can be printed and summarized using `print` and `summary` functions. The summary of an object `snp` includes both genotype and allele frequencies and the Hardy-Weinberg equilibrium test.

```
> summary(mySNP)
Genotypes:
    frequency percentage
T/T        92 0.58598726
C/T        53 0.33757962
C/C        12 0.07643312

Alleles:
  frequency percentage
T       237  0.7547771
C        77  0.2452229

HWE (p value): 0.28163925
```

An object of class `snp` may also be plotted using the `plot` function. Different types of plots may be obtained just by changing the argument `type`. Figure 1 shows two different plots for a given SNP. They have been obtained using the next instructions, where the arguments `label` and `col` are optional.

```
> # left figure
> plot(mySNP,label="snp10001",col="darkgreen")
> # right figure
> plot(mySNP,type=pie,label="snp10001",col=c("darkgreen","yellow","red"))
```

Other methods such as `reorder` are also implemented for an object of class `snp`. In that case, the argument `ref` determines whether the genotype with common allele is the reference or not.

```
> reorder(mySNP,ref="minor")
  [1] T/T T/T T/T C/T T/T T/T T/T T/T C/T T/T C/T C/C C/T T/T T/T T/T C/T T/T
 [19] T/T T/T T/T C/C C/T T/T T/T C/T T/T T/T T/T C/C T/T T/T C/T T/T C/T C/T
 [37] C/C C/T C/T T/T T/T T/T T/T C/T T/T C/C C/T C/T C/T T/T T/T C/T C/T T/T
 [55] T/T T/T T/T T/T C/T T/T T/T T/T T/T T/T T/T T/T T/T C/C T/T T/T C/T C/T T/T
 [73] T/T C/T T/T T/T T/T T/T T/T C/T C/T T/T T/T T/T C/C T/T T/T C/C C/C C/T
 [91] T/T T/T C/T T/T T/T T/T T/T T/T C/T T/T C/T C/T C/T T/T T/T C/C C/T T/T
[109] T/T C/T T/T C/T C/T T/T C/T C/T T/T T/T C/T C/T C/T T/T C/C T/T T/T T/T
[127] T/T T/T C/T T/T C/T C/C T/T C/T C/T T/T C/T T/T T/T C/T C/T C/T T/T T/T
[145] T/T T/T C/T T/T T/T T/T T/T C/T T/T C/T C/T C/T C/T
Levels: C/C C/T T/T
```

Now, we can see as the genotype `C/C` is the reference. In order to help other possible codifications of the genotypes the user is allowed to indicate which are their codes (for instance, 0,1, and 2 or "homozig1", "heteroz", "homozig2"). As an example:

```
> gg<-c("het","hom1","hom1","hom1","hom1","hom1","het","het","het",
+   "hom1","hom2","hom1","hom2")
> snp(gg,name.genotypes=c("hom1","het","hom2"))
 [1] A/B A/A A/A A/A A/A A/A A/B A/B A/B A/A B/B A/A B/B
Levels: A/A A/B B/B
```

1.2 The class `setupSNP`

Previous functions are useful for dealing with a unique SNP. However in association studies we are normally interested in analyzing a huge number of SNPs. Thus, to indicate which variables are SNPs in our data set we use the function called `setupSNP`. This function prepares the data for being analyzed using other function as we will illustrate later. The following instruction is used to create an object of class `setupSNP`.

```
> myData<-setupSNP(data=SNPs,colSNPs=6:40,sep="")
```

This function creates a data frame of class `"setupSNP"` where the variables indicated in the argument `colSNPs` are converted to class `"snp"`. This object has four additional attributes, called `"label.SNPs"`, `"colSNPs"`, `"gen.info"`, and `"whole"`. These attributes encode the information about the names and columns of SNPs, the genomic information of SNPs (chromosome and position) and whether a whole genome analysis is carried out.

In some occasions one may be interested in having the SNPs sorted by chromosomes and genomic positions. To do so, the argument `sort` must be set to `TRUE`. In addition, we must indicate the genomic information through the argument `info` as follows:

```
> myData.o<-setupSNP(SNPs, colSNPs=6:40, sort=TRUE,
+  info=SNPs.info.pos, sep="")
```

where the information of `SNPs.info.pos` looks like this:

```
         snp  chr     pos
1   snp10001 Chr1 2987398
2   snp10002 Chr1 1913558
3   snp10003 Chr1 1982067
4   snp10004 Chr1  447403
5   snp10005 Chr1 2212031
6   snp10006 Chr1 2515720
7   snp10007 Chr1 1306743
8   snp10008 Chr1 2063658
9   snp10009 Chr1 3403359
```

```
10 snp100010 Chr1 1857134
11 snp100011 Chr2 2439115
12 snp100012 Chr2 1978467
13 snp100013 Chr2 1641528
14 snp100014 Chr2 3852933

...
```

The generic function `labels` may be used to obtain the names of SNPs for an object of class "setupSNP".

```
> labels(myData)
 [1] "snp10001"  "snp10002"  "snp10003"  "snp10004"  "snp10005"  "snp10006"
 [7] "snp10007"  "snp10008"  "snp10009"  "snp100010" "snp100011" "snp100012"
[13] "snp100013" "snp100014" "snp100015" "snp100016" "snp100017" "snp100018"
[19] "snp100019" "snp100020" "snp100021" "snp100022" "snp100023" "snp100024"
[25] "snp100025" "snp100026" "snp100027" "snp100028" "snp100029" "snp100030"
[31] "snp100031" "snp100032" "snp100033" "snp100034" "snp100035"
```

An object of class "setupSNP" can also be summarized obtaining information about alleles, major frequency allele, HWE test and missing genotypes as follows:

```
> summary(myData)
          alleles major.allele.freq HWE      missing (%)
snp10001  T/C      75.5             0.281639  0.0
snp10002  C/A      72.0             0.004945  0.0
snp10003  G       100.0             -         8.3
snp10004  G       100.0             -         0.6
snp10005  G/A      75.8             0.008020  0.0
snp10006  A       100.0             -         0.0
snp10007  C       100.0             -         0.0
snp10008  C/G      80.3             0.137802  0.0
snp10009  A/G      71.5             0.002848  0.6
snp100010 T       100.0             -         6.4
snp100011 G/C      98.7             0.019139  0.0
snp100012 G/C      76.1             0.013399  1.3
snp100013 A/G      81.7             0.025588  7.6
snp100014 A/C      58.2             1.000000  2.5
snp100015 G/A      95.9             -         0.0
snp100016 G       100.0             -         3.2
snp100017 T/C      70.0             0.000518  1.3
snp100018 T/C      69.9             0.000498  0.6
snp100019 C/G      55.7             0.746284  0.0
snp100020 G/A      80.6             0.125355  0.0
snp100021 G       100.0             -         0.0
snp100022 A       100.0             -         0.6
snp100023 T/A      71.4             0.002842  1.9
snp100024 T/C      74.7             0.092210  0.6
snp100025 C       100.0             -         0.0
snp100026 G       100.0             -         0.6
snp100027 C/G      70.3             0.000896  1.3
snp100028 C/T      55.1             0.419687  0.6
snp100029 G/A      75.6             0.048709  0.6
snp100030 A       100.0             -         0.0
snp100031 T       100.0             -        35.0
snp100032 A/G      55.8             0.258909  0.6
```

```
snp100033 A/G      54.9             0.326373  3.2
snp100034 T/C      75.6             0.048709  0.6
snp100035 T       100.0             -         7.0
```

After having and object of class "setupSNP" we may summarize and plot a given SNP using the generic function `plot` as follows:

```
> plot(myData,which=20)
snp100020
Genotypes:
    frequency percentage
G/G       105  66.878981
A/G        43  27.388535
A/A         9   5.732484

Alleles:
  frequency percentage
G       253   80.57325
A        61   19.42675

HWE (p value): 0.1253547
```

where the argument `which` indicates the position of the SNP we are interested in looking at.

1.3 Missing data

We may have a look at the information we have for each SNPs using `plotMissing` function. This function requires the data to be an object of class `setupSNP`. The top plot in Figure 2 shows the missing information for `SNPs` data set. This figure may be obtained typing:

```
> plotMissing(myData)
```

If we execute `plotMissing(myData.o)`, as the `myData.o` is an object of class `setupSNP` with the genomic information, the `plotMissing` function gives a plot including that information (bottom plot in Figure 2)

1.4 Hardy-Weinberg equilibrium (HWE)

Now, we are interested in checking Hardy-Weinberg equilibrium for a set of SNPs. Here, we take advantage of having an object-oriented program since the function compute HWE test for all variables of class `snp` included in an object of class `setupSNP`

```
> res<-tableHWE(myData)
> res
          HWE (p value) flag
snp10001  0.2816
snp10002  0.0049        <-
snp10003  -
snp10004  -
snp10005  0.0080        <-
snp10006  -
snp10007  -
snp10008  0.1378
snp10009  0.0028        <-
snp100010 -
snp100011 0.0191        <-
```

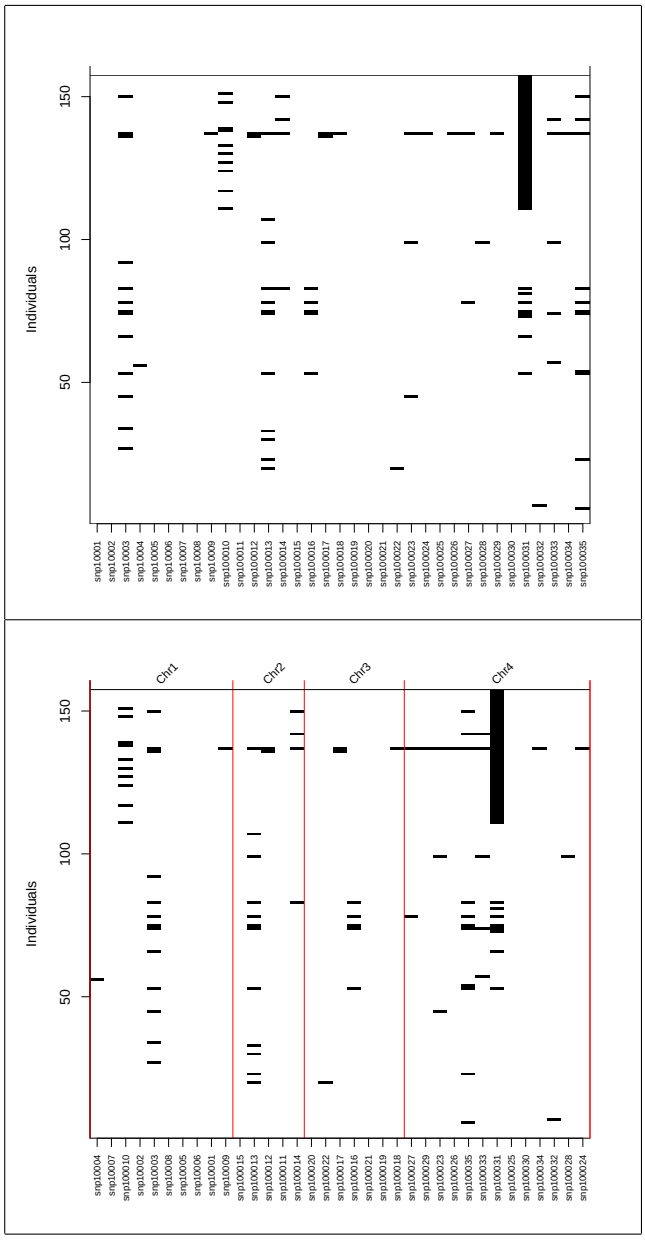

Figure 2: Missing information for the genotyped SNPs. The bottom figure is the same as the top one with the SNPs sorted by genomic position at each chromosome.

```
snp100012 0.0134        <-
snp100013 0.0256        <-
snp100014 1.0000
snp100015 -
snp100016 -
snp100017 0.0005        <-
snp100018 0.0005        <-
snp100019 0.7463
snp100020 0.1254
snp100021 -
snp100022 -
snp100023 0.0028        <-
snp100024 0.0922
snp100025 -
snp100026 -
snp100027 0.0009        <-
snp100028 0.4197
snp100029 0.0487        <-
snp100030 -
snp100031 -
snp100032 0.2589
snp100033 0.3264
snp100034 0.0487        <-
snp100035 -
```

The column indicated by `flag` shows those SNPs that are statistically significant at level 0.05. This significance level may be changed using the argument `sig` in the print function (e.g. `print(myData, sig=0.0001)`). The number of decimals may also be changed using the `digits` parameter. A stratified analysis may also be performed using the argument `strata` as follows:

```
> res<- tableHWE(myData,strata=myData$sex)
> res
          all.groups   Male Female
snp10001      0.2816 0.3941 0.7388
snp10002      0.0049 0.1660 0.0075
snp10003           -      -      -
snp10004           -      -      -
snp10005      0.0080 0.2755 0.0257
snp10006           -      -      -
snp10007           -      -      -
snp10008      0.1378 0.5078 0.2342
snp10009      0.0028 0.0992 0.0075
snp100010          -      -      -
snp100011     0.0191      - 0.0184
snp100012     0.0134 0.2761 0.0255
snp100013     0.0256 0.1206 0.2051
snp100014     1.0000 0.8101 0.6456
snp100015          -      -      -
snp100016          -      -      -
snp100017     0.0005 0.0304 0.0068
snp100018     0.0005 0.0304 0.0066
snp100019     0.7463 1.0000 0.5012
snp100020     0.1254 0.5078 0.2141
snp100021          -      -      -
```

```
snp100022         -      -      -
snp100023    0.0028 0.0972 0.0123
snp100024    0.0922 0.1551 0.5197
snp100025         -      -      -
snp100026         -      -      -
snp100027    0.0009 0.0304 0.0123
snp100028    0.4197 1.0000 0.2619
snp100029    0.0487 0.0772 0.5065
snp100030         -      -      -
snp100031         -      -      -
snp100032    0.2589 0.8170 0.1834
snp100033    0.3264 0.8139 0.2619
snp100034    0.0487 0.0772 0.5065
snp100035         -      -      -
```

2 Whole genome association studies

2.1 The class `WGassociation`

After an initial inspection of the data (genotyping and allele frequencies, missing data, and HWE test), Whole Genome `association` studies (objects of class `"WGassociation"`) can be analyzed with `SNPassoc` using `WGassociation` function. An object of class `setupSNP` is needed. Normally, when we are performing genome-wide association studies we may analyze hundreds of SNPs in different chromosomes. So, in these kind of studies genetic information will be needed. In addition, we will need to indicate to the object of class `"setupSNP"` that a whole genome analysis will be carried out. Let us illustrate this procedure using a real data set. We have obtained information about 10,000 SNPs from the HapMap project (http://www.hapmap.org) belonging to all chromosomes. We were interested in comparing the genotype frequencies for all variants among European population (CEU) and Yoruba (YRI). The data set containing this information is available in a data frame called `HapMap`. The genomic information (names of SNPs, chromosomes and genetic position) is also available in a data frame called `HapMap.SNPs.pos`. Both objects can be loaded typing `data(HapMap)`. The required object of class `setupSNP` is then created by executing:

```
> data(HapMap)
> myDat.HapMap<-setupSNP(HapMap, colSNPs=3:9307, sort = TRUE,
+   info=HapMap.SNPs.pos, sep="")
```

After obtaining the object of class `setupSNP`, the association analysis is then performed typing: (NOTE: `resHapMap` object is loaded after executing `data(HapMap)`. So it is not necessary to execute the next instruction)

```
> resHapMap<-WGassociation(group, data=myDat.HapMap, model="log-add")
```

where `group` is a factor with levels CEU and YRI. In this example`WGassociation` will fit individual logistic regression models to each of the variables class "snp" provided in the "setupSNP" data object `myDat`. If we need an analysis adjusted for covariates, these can be indicated using a model formula. For example, to adjust the association of each SNP for age and sex we would use `group age+sex`. Analysis assuming different genetic models may be obtained with the argument `model` (codominant, dominant, recessive, overdominant, log-additive or all). Since a genome-wide association analysis may be very time consuming, we recommend the user to change this argument and carry out the association assuming only one genetic model in a preliminary step (in our example `"log-additive"`). The value returned by `WGassociation` is an object of class `"WGassociation"`. It can be stored, plotted, and inspected using the methods for the generic operations: `plot`, `print` and `summary`. The function `summary` provides, for each SNP and genetic model, a cross tabulation with numbers

and percentages, ORs (or mean differences in quantitative traits), 95% confidence intervals, the p-value for the likelihood ratio test of association, and the Akaike information criteria. Figure 3 shows a plot for a whole genome analysis assuming a log-additive mode of inheritance. This plot may be obtained typing `plot(resHapMap)`. The argument `cutPval` may be used for changing threshold of those SNPs that are statistically significant. As an example, if we consider a p-value of 5×10^{-8}, `cuPval` should be set to `cutPval=c(0,5e-08,1)`. This figure is obtained by default when more than 10 chromosomes (or genes) are analyzed. The user may obtain other kind of plot (Figure 4) just by changing the argument `whole=FALSE`.

```
> plot(resHapMap, whole=FALSE, print.label.SNPs = FALSE)
```

When there is not information about chromosomes but about genes, a similar plot may also be obtained setting both `sort.chromosome` and `centromere` to `FALSE`.

The information given in Figure 3 may be summarized using the function `summary` as follows:

```
> summary(resHapMap)
      SNPs (n) Genot error (%) Monomorphic (%) Significant* (n)  (%)
chr1       796             3.8            18.6               163 20.5
chr2       789             4.2            13.9               161 20.4
chr3       648             5.2            13.0               132 20.4
chr4       622             6.3            17.7               104 16.7
chr5       587             4.4            14.7               118 20.1
chr6       556             4.1            16.9               101 18.2
chr7       515             5.8            15.7                96 18.6
chr8       476             4.4            13.7                99 20.8
chr9       450             6.4            15.3                98 21.8
chr10      440             2.7            18.2                99 22.5
chr11      437             7.1            17.6                75 17.2
chr12      431             6.5            16.7                79 18.3
chr13      371             2.7            13.2                75 20.2
chr14      346             7.8            15.0                60 17.3
chr15      326             4.6            12.0                76 23.3
chr16      288             4.5            17.7                61 21.2
chr17      256             6.2            17.2                60 23.4
chr18      247             4.9            17.8                38 15.4
chr19      207             6.3            18.4                41 19.8
chr20      203             3.0            30.5                34 16.7
chr21      153             6.5            14.4                28 18.3
chr22      161             3.7            25.5                28 17.4

 *Number of statistically significant associations at level 1e-06
```

As we will later illustrate, the `WGassociation` function computes for each SNP: ORs, confidence intervals, p values from likelihood ratio test (LRT) and AIC. However, in many occasions the researcher is only interested in knowing those SNPs that are statistically significant at a given level (i.e. p value from LRT). Thus, in order to save computing time we have programmed an alternative function, called `scanWGassociation`, for analyzing whole genome data sets when p values are the only required. This function also returns an object of class "WGasscociation". So all methods and functions implemented for objects of class "WGasscociation" may also be used for inspecting those results obtained using `scanWGassociation`. The HapMap data set may also be analyzed using:

```
> resHapMap.scan<-scanWGassociation(group, data=myDat.HapMap, model="log-add")
Be patient. The program is computing ...
The program took 1.87 seconds
```

which is extremely less time-consuming than to use the `WGassociation` function.

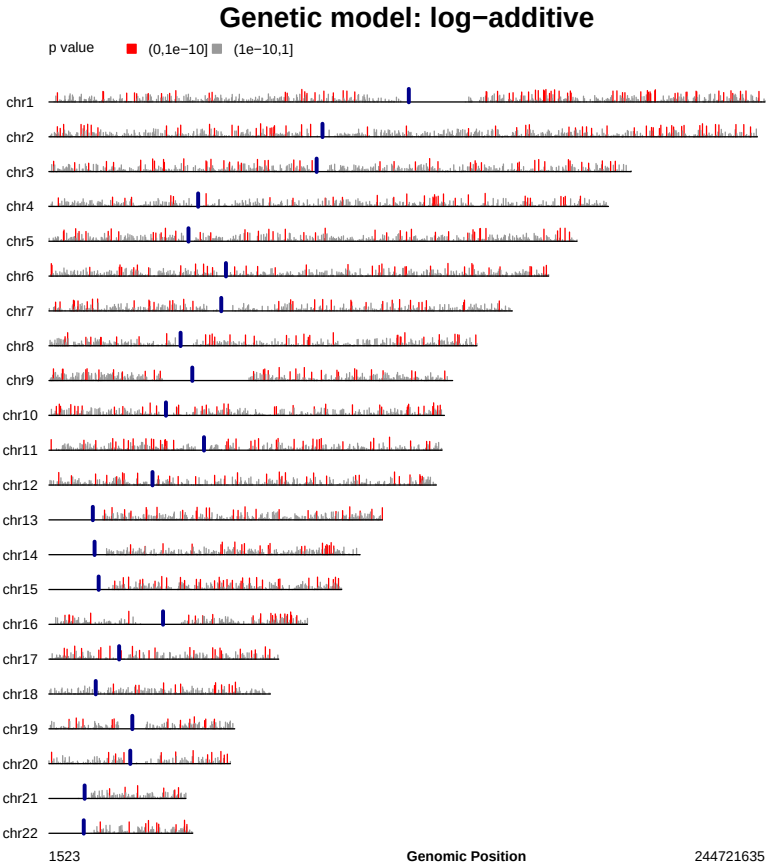

Figure 3: Results of `WGassociation` for the HapMap data set. The -log p values for a whole genome analysis assuming a log-additive genetic model are showed for each chromosome. The statistically significant associations at level 10^{-15} are plotted in red, while the other associations are in gray. Blue lines indicate the centromeres.

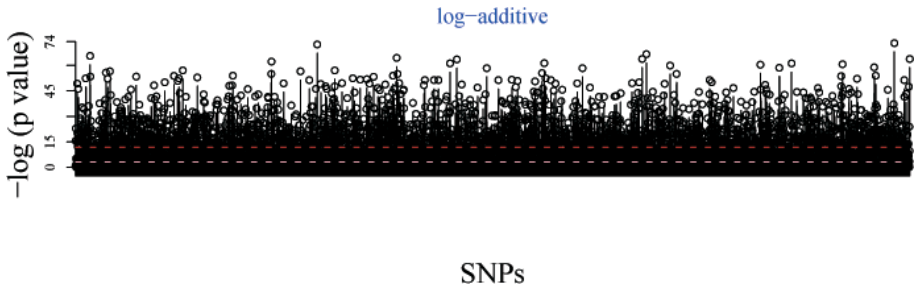

Figure 4: Results of `WGassociation` for the HapMap data set. The -log p values for a whole genome analysis assuming a log-additive genetic model are showed. The statistically significant associations at nominal level (pink line) and at Bonferroni corrected level (red line) are also indicated.

2.2 Permutation and related tests

Permutation test is a widely-used method to compute significance level. In a whole genome association study the problem is that genotypes are correlated and such correlation might be consider to be successful. The standard procedure is to permute trait values among individuals while keeping their genotypes fixed. The minimum p-value is obtained at each permutation to estimate its empirical distribution and to get the significance level. Another possibility is to obtain the significance level assuming that minimum p-values are distributed as a Beta distribution [Dudbridge and Koeleman, 2004].

Dudbridge et al. (2006) stated that the accuracy of permutation test can be improved by noting that the minimum P-value (and other statistics such as sum statistic or truncated product) can be regarded as the extreme value of a large number of observations. Thus, they propose to use the extreme value distribution to obtain more accurate significance levels [Dudbridge et al, 2006].

SNPassoc performs the permutation test (we must say that this procedure is ONLY available for binary traits) using the function `scanWGassociation`. Since this procedure is extremely time-consuming, it has been implemented using FORTRAN routines which are called from R functions via a dll. The p values are obtained executing:

```
> resHapMap.perm<-scanWGassociation(group, data=myDat.HapMap,
+ model="log-add", nperm=1000)
Be patient. The program is computing ...
The program took 277.17 seconds
```

Then, the permutation test is performed typing:

```
> res.perm<- permTest(resHapMap.perm)
> print(res.perm)

Permutation test analysis (95% confidence level)
------------------------------------------------
Number of SNPs analyzed:  9305
Number of valid SNPs (e.g., non-Monomorphic and passing calling rate):  9305
P value after Bonferroni correction:  5.37e-06

P values based on permutation procedure:
P value from empirical distribution of minimum p values:  1.79e-05
P value assuming a Beta distribution for minimum p values:  1.931e-05
```

Figure 5 shows significance level from the permutation test which is easily obtained by writing:.

```
> plot(res.perm)
```

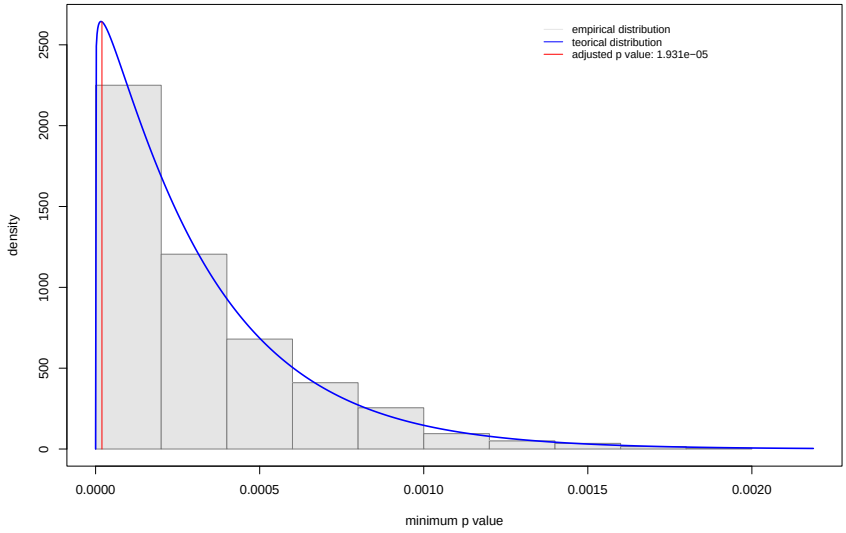

Figure 5: Empirical and theoretical distribution, assuming a Beta(1,α), for the minimum p values. Results obtained from a permutation test for HapMap data set. Red line indicates the adjusted significance level.

The rank truncated product [Dudbridge et al, 2006] is implemented in `permTest` function, just indicating `method="rtp"` and the number of the K most significant p-values.

```
> res.perm.rtp<- permTest(resHapMap.perm,method="rtp",K=20)
> print(res.perm.rtp)

Permutation test analysis (95% confidence level)
-------------------------------------------------
Number of SNPs analyzed:  9305
Number of valid SNPs (e.g., non-Monomorphic and passing calling rate):  9305
P value after Bonferroni correction:  5.37e-06

Rank truncated product of the K=20 most significant p-values:
Product of K p-values (-log scale):  947.2055
Significance:  <0.001
```

3 Medium/Large scale association studies

The preliminary step helps the researchers to identify a subset of SNPs with putative associations. In a following stage, those SNPs are retested in populations that have larger size. To identify the set of SNPs with association that are statistically significant at a given level (by default `1e-15`) the function `getSignificantSNPs` may be used as follows:

```
> getSignificantSNPs(resHapMap,chromosome=5)
$names
 [1] "rs6555568"  "rs4702723"  "rs4866272"  "rs7720894"  "rs6452430"
 [6] "rs10067664" "rs6880750"  "rs267030"   "rs179194"   "rs809039"
[11] "rs1015565"  "rs6871275"  "rs1864998"  "rs263890"   "rs11955678"
[16] "rs1702380"  "rs1106986"

$column
 [1] 6726 6742 6807 6927 6985 7022 7099 7101 7107 7123 7143 7157 7204 7260 7268
[16] 7277 7290
```

where `resHapMap` is the object of class `WGassociation` previously fitted, which contains the results of a whole genome analysis, and the argument `chromosome` indicates the chromosome from which the significant SNPs are obtained. SNPs associated with the outcome at different level of signification may be obtained modifying the argument `sig` (e.g. `sig=5e-08` for a significant level of 5×10^{-8})

After determining this set of SNPs, we are normally interested in not only assessing association between dependent variable and SNPs but also in further investigating how the SNPs and the disease are associated (i.e. which is the mode of inheritance). Thus, in this second step, other functions included in `SNPassoc` package may be used. To illustrate them we are using the SNPs data set presented in Section 1.

3.1 Association with a single SNP

3.1.1 Crude analysis

We may assess the association between a given SNP and the outcome using `association` function. To do so, it is necessary to incorporate in the model a variable of class `snp`. There are two different possibilities. The first one is to use the function `snp` in the formula as follows:

```
> association(casco~snp(snp10001,sep=""), data=SNPs)

SNP: snp10001, sep = ""  adjusted by:
              0    %   1    %   OR lower upper p-value   AIC
Codominant
T/T          24 51.1  68 61.8 1.00              0.1323 193.6
C/T          21 44.7  32 29.1 0.54  0.26  1.11
C/C           2  4.3  10  9.1 1.76  0.36  8.64
Dominant
T/T          24 51.1  68 61.8 1.00              0.2118 194.1
C/T-C/C      23 48.9  42 38.2 0.64  0.32  1.28
Recessive
T/T-C/T      45 95.7 100 90.9 1.00              0.2715 194.4
C/C           2  4.3  10  9.1 2.25  0.47 10.69
Overdominant
T/T          26 55.3  78 70.9 1.00              0.0613 192.1
T/T-C/C      21 44.7  32 29.1 0.51  0.25  1.03
log-Additive
0,1,2        47 29.9 110 70.1 0.87  0.51  1.47  0.5945 195.4
```

The another possibility, which is recommended, is to use an object of class `setupSNPs` as the data frame. Thus, it is not necessary to indicate which is the variable of class `snp` since the object has this information as we have previously illustrated. Thus, we should type:

```
> myData<-setupSNP(data=SNPs,colSNPs=6:40,sep="")
> association(casco~snp10001, data=myData)

SNP: snp10001  adjusted by:
              0    %   1    %   OR lower upper p-value   AIC
Codominant
T/T          24 51.1  68 61.8 1.00              0.1323 193.6
C/T          21 44.7  32 29.1 0.54  0.26  1.11
C/C           2  4.3  10  9.1 1.76  0.36  8.64
Dominant
T/T          24 51.1  68 61.8 1.00              0.2118 194.1
C/T-C/C      23 48.9  42 38.2 0.64  0.32  1.28
Recessive
T/T-C/T      45 95.7 100 90.9 1.00              0.2715 194.4
C/C           2  4.3  10  9.1 2.25  0.47 10.69
Overdominant
T/T          26 55.3  78 70.9 1.00              0.0613 192.1
T/T-C/C      21 44.7  32 29.1 0.51  0.25  1.03
log-Additive
0,1,2        47 29.9 110 70.1 0.87  0.51  1.47  0.5945 195.4
```

By default this function calculates the association between the SNP and the dependent variable (left side of the formula) under five different genetic models. The argument `model` may be used for analyzing only some of them. Let's assume that we are only interested in analyzing codominant and log-additive models. In that case, the instructions are:

```
> association(casco~snp10001, data=myData, model=c("cod","log"))

SNP: snp10001  adjusted by:
              0    %  1    %   OR lower upper p-value   AIC
Codominant
T/T          24 51.1 68 61.8 1.00              0.1323 193.6
C/T          21 44.7 32 29.1 0.54  0.26  1.11
C/C           2  4.3 10  9.1 1.76  0.36  8.64
log-Additive
0,1,2        47 29.9 110 70.1 0.87  0.51  1.47  0.5945 195.4
```

The output is self-defined. Labels make reference to different genetic mode of inheritance. We notice that a quantitative trait may be analyzed using the same instruction just changing `casco` by a continuous variable (the user may try `protein` as an example). We highlight that it is not necessary to indicate whether the trait is quantitative because when a factor variable with two levels is written in the left side of the formula, a case-control study is performed. Anyway, the user may force to perform a quantitative analysis indicating that the argument `quantitative` is `TRUE`.

3.1.2 Adjusted analysis

Now, we can analyze this SNP adjusted by other covariates, such as gender or arterial blood pressure, as follows:

```
>  association(casco~sex+snp10001+blood.pre, data=myData)
```

```
SNP: snp10001  adjusted by: sex blood.pre
              0    %   1    %   OR lower upper p-value   AIC
Codominant
T/T          24 51.1  68 61.8 1.00             0.15410 195.8
C/T          21 44.7  32 29.1 0.55  0.26  1.14
C/C           2  4.3  10  9.1 1.74  0.35  8.63
Dominant
T/T          24 51.1  68 61.8 1.00             0.22859 196.1
C/T-C/C      23 48.9  42 38.2 0.65  0.32  1.31
Recessive
T/T-C/T      45 95.7 100 90.9 1.00             0.28494 196.4
C/C           2  4.3  10  9.1 2.22  0.46 10.70
Overdominant
T/T          26 55.3  78 70.9 1.00             0.07188 194.3
T/T-C/C      21 44.7  32 29.1 0.52  0.25  1.06
log-Additive
0,1,2        47 29.9 110 70.1 0.87  0.51  1.49 0.60861 197.3
```

3.1.3 Stratified analysis

We may also be interested in analyzing this SNP for two different populations (i.e. stratified analysis). Let us assume that we want to compute the same ORs for males and females. In this case `strata` function from `survival` package may be used as follows:

```
> association(casco~snp10001+blood.pre+strata(sex), data=myData)

       strata: sex=Male
SNP: snp10001  adjusted by: blood.pre
              0     %  1    %   OR lower upper p-value  AIC
Codominant
T/T          11  52.4 29 53.7 1.00             0.04070 90.3
C/T          10  47.6 17 31.5 0.63  0.22  1.80
C/C           0   0.0  8 14.8       0.00
Dominant
T/T          11  52.4 29 53.7 1.00             0.89492 94.7
C/T-C/C      10  47.6 25 46.3 0.93  0.34  2.57
Recessive
T/T-C/T      21 100.0 46 85.2 1.00             0.01740 89.1
C/C           0   0.0  8 14.8       0.00
Overdominant
T/T          11  52.4 37 68.5 1.00             0.18207 92.9
T/T-C/C      10  47.6 17 31.5 0.49  0.17  1.39
log-Additive
0,1,2        21  28.0 54 72.0 1.35  0.62  2.95 0.44244 94.1

       strata: sex=Female
SNP:  adjusted by:
              0    %  1    %   OR lower upper p-value   AIC
Codominant
T/T          13 50.0 39 69.6 1.00             0.3054 102.7
C/T          11 42.3 15 26.8 0.49  0.17  1.38
C/C           2  7.7  2  3.6 0.35  0.04  2.88
Dominant
T/T          13 50.0 39 69.6 1.00             0.1309 100.8
C/T-C/C      13 50.0 17 30.4 0.47  0.17  1.25
```

```
Recessive
T/T-C/T      24 92.3 54 96.4 1.00              0.4595 102.5
C/C           2  7.7  2  3.6 0.46  0.06  3.60
Overdominant
T/T          15 57.7 41 73.2 1.00              0.2290 101.6
T/T-C/C      11 42.3 15 26.8 0.54  0.19  1.47
log-Additive
0,1,2        26 31.7 56 68.3 0.54  0.24  1.20  0.1300 100.8
```

3.1.4 Subset analysis

In some occasions one may be interested in analyzing data only in a subset of individuals. This can be easily done using `subset` argument.

```
> association(casco~snp10001+blood.pre, data=myData,
+       subset=sex=="Male")

SNP: snp10001  adjusted by: blood.pre
              0     %  1    %   OR lower upper p-value  AIC
Codominant
T/T          11  52.4 29 53.7 1.00             0.04070 90.3
C/T          10  47.6 17 31.5 0.63  0.22  1.80
C/C           0   0.0  8 14.8       0.00
Dominant
T/T          11  52.4 29 53.7 1.00             0.89492 94.7
C/T-C/C      10  47.6 25 46.3 0.93  0.34  2.57
Recessive
T/T-C/T      21 100.0 46 85.2 1.00             0.01740 89.1
C/C           0   0.0  8 14.8       0.00
Overdominant
T/T          11  52.4 37 68.5 1.00             0.18207 92.9
T/T-C/C      10  47.6 17 31.5 0.49  0.17  1.39
log-Additive
0,1,2        21  28.0 54 72.0 1.35  0.62  2.95 0.44244 94.1
```

The same analyses may be performed for a quantitative trait replacing `casco` by a continuous variable such as `protein`. The only difference is that the output is obviously slightly different. As an example, let us suppose that we are interested in analyzing the effect of the SNP `snp10029` and protein levels for males and females, adjusted by arterial blood pressure. To do so, we should execute:

```
> association(log(protein)~snp100029+blood.pre+strata(sex), data=myData)

       strata: sex=Male
SNP: snp100029  adjusted by: blood.pre
              n    me      se      dif   lower    upper p-value   AIC
Codominant
G/G          42 10.64 0.07722  0.00000                  0.02949 136.4
A/G          23 10.51 0.11754 -0.13259 -0.4299  0.16474
A/A           9 10.07 0.31101 -0.56823 -0.9892 -0.14730
Dominant
G/G          42 10.64 0.07722  0.00000                  0.06801 138.1
A/G-A/A      32 10.39 0.12369 -0.25505 -0.5290  0.01887
Recessive
G/G-A/G      65 10.59 0.06486  0.00000                  0.01204 135.2
```

```
A/A           9 10.07 0.31101 -0.52112 -0.9279 -0.11434
Overdominant
G/G          51 10.54 0.08759  0.00000                  0.83495 141.4
G/G-A/A      23 10.51 0.11754 -0.03186 -0.3316  0.26787
log-Additive
0,1,2                        -0.24135 -0.4315 -0.05119 0.01286 135.3

       strata: sex=Female
SNP:  adjusted by:
              n     me      se     dif   lower    upper   p-value   AIC
Codominant
G/G          52 10.607 0.07686  0.0000                  0.0001702 175.3
A/G          25 10.326 0.16002 -0.2713 -0.5961  0.05359
A/A           5  9.286 0.52398 -1.2954 -1.9214 -0.66947
Dominant
G/G          52 10.607 0.07686  0.0000                  0.0074509 182.7
A/G-A/A      30 10.153 0.17075 -0.4402 -0.7625 -0.11777
Recessive
G/G-A/G      77 10.516 0.07436  0.0000                  0.0001499 176.1
A/A           5  9.286 0.52398 -1.2053 -1.8284 -0.58218
Overdominant
G/G          57 10.491 0.09551  0.0000                  0.3843038 189.0
G/G-A/A      25 10.326 0.16002 -0.1553 -0.5052  0.19457
log-Additive
0,1,2                         -0.4679 -0.7154 -0.22048 0.0002103 176.7
```

3.1.5 Interaction analysis

An interaction term, generally one SNP with a categorical covariate, may be included in the formula. Then, the ORs (or mean differences if a quantitative trait is analyzed) and their 95% confidence intervals are expressed with respect to the non variant genotype and the first category of the covariate. The other two tables show the ORs and their 95% confidence intervals for both marginal models. P values for interaction and trend are also showed in the output. We use the `print` function to obtain a nicer output to read (less decimals using `digit` argument).

```
> ans<-association(log(protein)~snp10001*sex+blood.pre, data=myData,
+ model="codominant")
> print(ans,dig=2)

      SNP: snp10001  adjusted by: blood.pre
 Interaction
---------------------
       Male        dif lower upper    Female          dif lower upper
T/T 40   11 0.08  0.00    NA    NA 52   10.6 0.079 -0.026 -0.29  0.24
C/T 27   11 0.10 -0.13 -0.45  0.19 26   10.2 0.184 -0.472 -0.79 -0.15
C/C  8   10 0.35 -0.64 -1.13 -0.14  4    9.8 0.286 -0.887 -1.56 -0.22

p interaction: 0.36051

 sex within snp10001
---------------------
T/T
        n me    se    dif lower upper
Male   40 11 0.080  0.000    NA    NA
```

```
Female 52 11 0.079 -0.026 -0.29  0.24

C/T
        n me   se   dif lower  upper
Male   27 11 0.10  0.00    NA     NA
Female 26 10 0.18 -0.34 -0.69 0.0086

C/C
       n   me   se   dif lower upper
Male   8 10.0 0.35  0.00    NA    NA
Female 4  9.8 0.29 -0.25  -1.0  0.53

p trend: 0.26575

 snp10001 within sex
---------------------
Male
     n me   se   dif lower upper
T/T 40 11 0.08  0.00    NA    NA
C/T 27 11 0.10 -0.13 -0.45  0.19
C/C  8 10 0.35 -0.64 -1.13 -0.14

Female
     n   me    se   dif lower upper
T/T 52 10.6 0.079  0.00    NA    NA
C/T 26 10.2 0.184 -0.45 -0.75 -0.14
C/C  4  9.8 0.286 -0.86 -1.52 -0.20

p trend: 0.36051
```

The mode of inheritance may be changed using the `model` argument. We may also obtain an interaction table for two SNPs using the command (notice that one of the snps migth be converted to factor)

```
> ans<-association(log(protein)~snp10001*factor(recessive(snp100019))
+        +blood.pre, data=myData, model="codominant")
> print(ans,dig=2)

      SNP: snp10001  adjusted by: blood.pre
 Interaction
---------------------
       G/G-C/G         dif lower  upper    C/C         dif lower upper
T/T 60      11 0.063  0.00    NA     NA 32  11 0.11 -0.038 -0.32  0.24
C/T 53      10 0.106 -0.30 -0.54 -0.053  0   0 0.00     NA    NA    NA
C/C 12      10 0.244 -0.72 -1.13 -0.313  0   0 0.00     NA    NA    NA

p interaction: NA

 factor(recessive(snp100019)) within snp10001
---------------------
T/T
         n me    se    dif lower upper
G/G-C/G 60 11 0.063  0.000    NA    NA
C/C     32 11 0.112 -0.038 -0.32  0.24
```

```
C/T
         n me   se dif lower upper
G/G-C/G 53 10 0.11   0    NA    NA
C/C      0  0 0.00  NA    NA    NA

C/C
         n me   se dif lower upper
G/G-C/G 12 10 0.24   0    NA    NA
C/C      0  0 0.00  NA    NA    NA

p trend: NA

 snp10001 within factor(recessive(snp100019))
---------------------
G/G-C/G
     n me    se   dif lower  upper
T/T 60 11 0.063  0.00    NA     NA
C/T 53 10 0.106 -0.30 -0.54 -0.053
C/C 12 10 0.244 -0.72 -1.13 -0.313

C/C
     n me   se dif lower upper
T/T 32 11 0.11   0    NA    NA
C/T  0  0 0.00  NA    NA    NA
C/C  0  0 0.00  NA    NA    NA

p trend: NA
```

3.2 Multiple tests

After that, we can carry out the same analysis for several SNPs. To do so we may also use the `WGassociation` function as in the case of a whole genome analysis. In that case, the `WGassociation` function will take into account that the object `myData` of class `"setupSNP"` will have the attribute `whole` equal `FALSE`. As an example, this fact will be important when method `plot` is used.

Here we notice that if we are interested in further analyzing HapMap data set we could take advantage of the previous analysis as follows. First, we create an object of class `"setupSNP"` with those SNPs with putative association typing:

```
> sigSNPs<-getSignificantSNPs(resHapMap,"X",sig=5e-8)$column
> myDat2<-setupSNP(HapMap, colSNPs=sigSNPs, sep="")
```

Then, the association analysis in this second step would be:

```
> resHapMap2<-WGassociation(group~1, data=myDat2)
> plot(resHapMap2,cex=0.8)
```

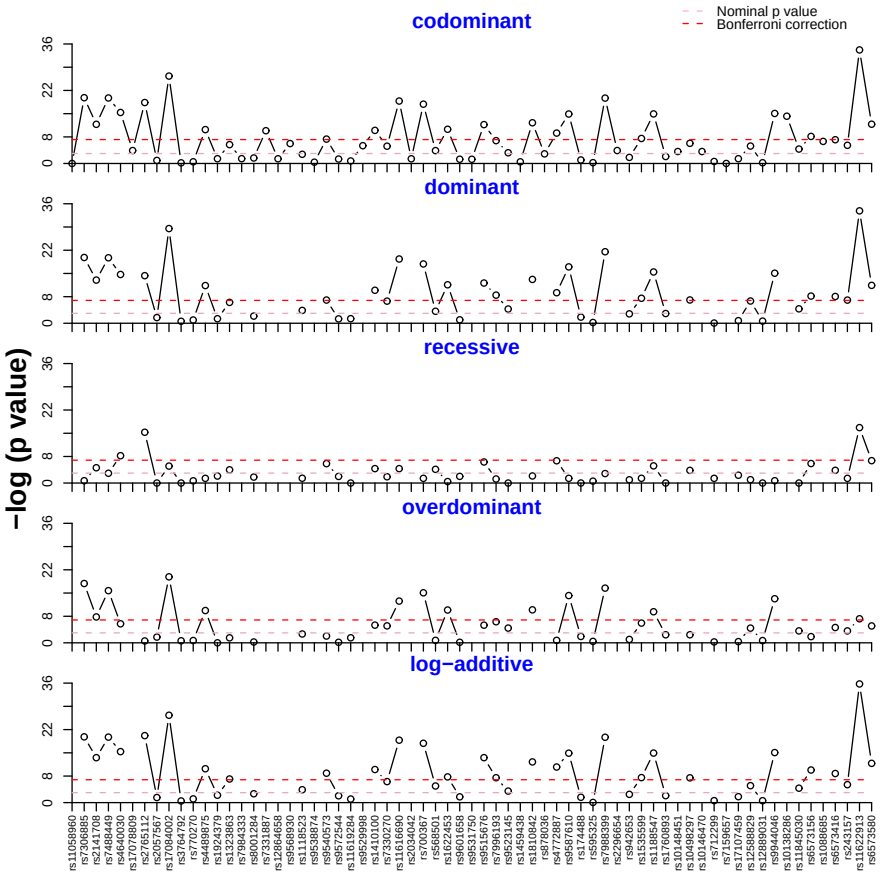

Figure 6: Results of `WGassociation` for the HapMap data set. The -log p values for a whole genome analysis for chromosome 5 and a set of SNPs with putative associations are showed for each mode of inheritance.

Figure 6 shows the p values (in -log scale) for the association analysis in the second step after selecting a given chromosome (5 in our case) and those SNPs with putative associations. We highlight that a different plot is obtained from that obtained when a whole genome analysis is carried out (Figure 3).

Let us turn to the example given in the SNPs data set where a moderate number of SNPs were analyzed. First, we recall how to create the object of class `"setupSNP"`

```
> myData<-setupSNP(SNPs, colSNPs=6:40, sep="")
```

The same object including genomic order:

```
> myData<-setupSNP(SNPs, colSNPs=6:40, sep="")
> myData.o<-setupSNP(SNPs, colSNPs=6:40, sort=TRUE,
+ info=SNPs.info.pos, sep="")
```

The association analysis is then performed as follows:

```
> ans<-WGassociation(protein~1,data=myData.o)
```

The function `WGassociation` carries out the same analyses as `association` does. The only difference is that we do not need to give any variable of class `snp` since the function performs the association analysis for all variables of class `snp` given in the `data` argument.

This argument is required an it has to be an object of class `setupSNP`. This function returns an object of class `WGassocition` as in the case of analyzing a whole genome data set. So, the "methods" implemented are the same we have previously mentioned. A short summary (only p values) is obtained as follows:

```
> ans
          comments    codominant dominant recessive overdominant log-additive
snp10004  Monomorphic -          -        -         -            -
snp10007  Monomorphic -          -        -         -            -
snp100010 Monomorphic -          -        -         -            -
snp10002  -           0.78525    0.93292  0.48600   0.87267      0.76807
snp10003  Monomorphic -          -        -         -            -
snp10008  -           0.20293    0.29843  0.08453   0.83628      0.13289
snp10005  -           0.63220    0.43763  0.50030   0.55340      0.37129
snp10006  Monomorphic -          -        -         -            -
snp10001  -           0.00492    0.00456  0.01491   0.12102      0.00114
snp10009  -           0.74695    0.87183  0.47708   0.68095      0.93605
snp100015 -           0.02484    -        -         -            -
snp100013 -           0.14592    0.09659  0.10819   0.38274      0.05278
snp100012 -           0.70516    0.58280  0.47821   0.72292      0.48889
snp100011 -           0.30259    0.12717  0.27118   0.28248      0.12583
snp100014 -           0.03531    0.01398  0.11743   0.26964      0.01143
snp100020 -           0.20671    0.31223  0.08453   0.86316      0.13932
snp100022 Monomorphic -          -        -         -            -
snp100017 -           0.70588    0.79091  0.45852   0.59896      0.99917
snp100016 Monomorphic -          -        -         -            -
snp100021 Monomorphic -          -        -         -            -
snp100019 -           0.02190    0.00674  0.14573   0.18974      0.00934
snp100018 -           0.75250    0.88674  0.47708   0.69475      0.92116
snp100027 -           0.92845    0.71446  0.94892   0.69917      0.75822
snp100029 -           0.00738    0.02052  0.00484   0.47493      0.00286
snp100023 -           0.77503    0.99543  0.48087   0.79853      0.82666
snp100026 Monomorphic -          -        -         -            -
snp100035 Monomorphic -          -        -         -            -
snp100033 -           0.01099    0.00397  0.06641   0.26365      0.00372
snp100031 Genot 65%   -          -        -         -            -
snp100025 Monomorphic -          -        -         -            -
snp100030 Monomorphic -          -        -         -            -
snp100034 -           0.00738    0.02052  0.00484   0.47493      0.00286
snp100032 -           0.01752    0.00585  0.09452   0.23811      0.00643
snp100028 -           0.01315    0.00444  0.08557   0.23352      0.00482
snp100024 -           0.00615    0.01654  0.00484   0.42319      0.00223
```

We notice that information about the quality of SNPs is also showed in the comments. As an example, we may observe that for the SNP called `snp100031` we obtain `Genot 65%` as a result, meaning that only 65% of individuals have information for this SNP. The percentage of genotyping for those SNPs that we are interested in including in the analysis is controlled by the argument `GenoRate`, which defaults to 80%. Previous analysis corresponds to a crude analysis. Adjusted results may be obtained replacing $\sim$ `1` by $\sim$ `age+sex`, if we are interested in obtaining results adjusted by `age` and `sex`. These results may be easily exported to LaTeX using `latex` function from `Hmisc` package as follows (Table 1). We notice that the information about p values may easily obtained using the function `pvalues`.

```
> library(Hmisc)
> SNP<-pvalues(ans)
> out<-latex(SNP,file="c:/temp/ans1.tex", where="'h",
+  caption="Summary of case-control study for SNPs data set.",
+ center="centering", longtable=TRUE, na.blank=TRUE,
+ size="scriptsize", collabel.just=c("c"), lines.page=50,
+ rownamesTexCmd="bfseries")
```

Table 1: Summary of case-control study for SNPs data set.

SNP	comments	codominant	dominant	recessive	overdominant	log-additive
snp10001	NA	0.0049	0.0046	0.0149	0.1210	0.0011
snp10002	NA	0.7853	0.9329	0.4860	0.8727	0.7681
snp10003	Monomorphic					
snp10004	Monomorphic					
snp10005	NA	0.6322	0.4376	0.5003	0.5534	0.3713
snp10006	Monomorphic					
snp10007	Monomorphic					
snp10008	NA	0.2029	0.2984	0.0845	0.8363	0.1329
snp10009	NA	0.7469	0.8718	0.4771	0.6810	0.9361
snp100010	Monomorphic					
snp100011	NA	0.3026	0.1272	0.2712	0.2825	0.1258
snp100012	NA	0.7052	0.5828	0.4782	0.7229	0.4889
snp100013	NA	0.1459	0.0966	0.1082	0.3827	0.0528
snp100014	NA	0.0353	0.0140	0.1174	0.2696	0.0114
snp100015	NA	0.0248				
snp100016	Monomorphic					
snp100017	NA	0.7059	0.7909	0.4585	0.5990	0.9992
snp100018	NA	0.7525	0.8867	0.4771	0.6947	0.9212
snp100019	NA	0.0219	0.0067	0.1457	0.1897	0.0093
snp100020	NA	0.2067	0.3122	0.0845	0.8632	0.1393
snp100021	Monomorphic					
snp100022	Monomorphic					
snp100023	NA	0.7750	0.9954	0.4809	0.7985	0.8267
snp100024	NA	0.0062	0.0165	0.0048	0.4232	0.0022
snp100025	Monomorphic					
snp100026	Monomorphic					
snp100027	NA	0.9285	0.7145	0.9489	0.6992	0.7582
snp100028	NA	0.0132	0.0044	0.0856	0.2335	0.0048
snp100029	NA	0.0074	0.0205	0.0048	0.4749	0.0029
snp100030	Monomorphic					
snp100031	Genot 65%					
snp100032	NA	0.0175	0.0059	0.0945	0.2381	0.0064
snp100033	NA	0.0110	0.0040	0.0664	0.2637	0.0037
snp100034	NA	0.0074	0.0205	0.0048	0.4749	0.0029
snp100035	Monomorphic					

The `WGstats` function returns the same analyses as in the case of analyzing a single SNP at time but for each of the SNPs included in the object of class `"setupSNP"`.

```
>  WGstats(ans,dig=5)
$snp10004
[1] "Monomorphic"

$snp10007
[1] "Monomorphic"

$snp100010
[1] "Monomorphic"

$snp10002
              n    me     se     dif   lower   upper p-value    AIC
Codominant
C/C           74 42876 2890.1    0.00                 0.78525 3612.5
A/C           78 42740 2575.9 -135.77 -376.13  104.59
A/A            5 50262 6879.3 7385.64 6701.24 8070.05
Dominant
C/C           74 42876 2890.1    0.00                 0.93292 3610.9
A/C-A/A       83 43193 2456.3  317.33   80.92  553.73
Recessive
C/C-A/C      152 42806 1924.1    0.00                 0.48600 3610.5
A/A            5 50262 6879.3 7455.31 6784.29 8126.34
Overdominant
C/C           79 43343 2741.8    0.00                 0.87267 3610.9
C/C-A/A       78 42740 2575.9 -603.22 -839.22 -367.21
log-Additive
0,1,2                         996.48  784.59 1208.36 0.76807 3610.9

.....

$snp100024
              n    me     se      dif    lower    upper   p-value    AIC
```

```
Codominant
T/T           91 46651 2338.8      0.0                    0.0061525 3580.0
C/T           51 40730 3423.4  -5920.9  -6172.2  -5669.6
C/C           14 26373 5559.6 -20277.8 -20690.3 -19865.4
Dominant
T/T           91 46651 2338.8      0.0                    0.0165390 3582.3
C/T-C/C       65 37638 3013.4  -9013.1  -9249.0  -8777.3
Recessive
T/T-C/T      142 44525 1946.1      0.0                    0.0048407 3580.2
C/C           14 26373 5559.6 -18151.3 -18555.3 -17747.3
Overdominant
T/T          105 43948 2252.7      0.0                    0.4231868 3587.4
T/T-C/C       51 40730 3423.4  -3217.2  -3469.1  -2965.3
log-Additive
0,1,2                         -8554.0  -8729.4  -8378.5 0.0022318 3578.8
```

When the attribute `whole` is FALSE, the function `plot` gives a different plot from that obtained in a whole genome analysis. Since we are normally interested in analyzing several modes of inheritance at the sime time, the p values (in -log scale) are plotted for each genetic model. Figure 7 shows the results in Table 1.

```
> plot(ans)
Warning: Any SNP is statistically significant after
                      Bonferroni Correction under codominant model
Warning: Any SNP is statistically significant after
                      Bonferroni Correction under dominant model
Warning: Any SNP is statistically significant after
                      Bonferroni Correction under recessive model
Warning: Any SNP is statistically significant after
                      Bonferroni Correction under overdominant model
```

Warnings showed after plotting an object of class `WGassociation` indicate how many SNPs are statistically significant after Bonferroni correction.

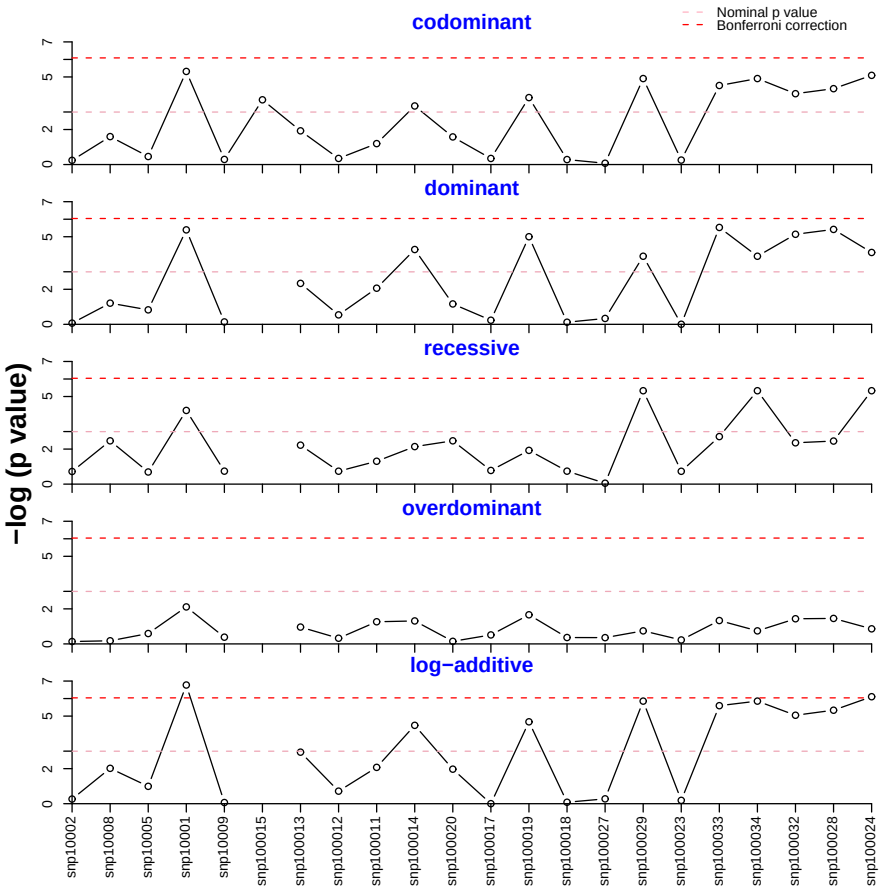

Figure 7: Results of `WGassociation` for the SNPs data set. The -log p values from likelihood ratio test for each SNPs are showed for each genetic model. The horizontal dotted lines indicate two different thresholds. One of them based on Bonferroni correction (red line), and another one in the nominal p-value wich is set equal to 0.05 (pink line).

3.2.1 Bonferroni correction

We may be interested in having these p values adjusted by the number of tests that we have carried out. One simple way to address this problem is using Bonferroni correction. As an example, let us obtain the SNPs that are statistically significant at 0.05 level after correcting by the number of tests and assuming a codominant model.

```
> Bonferroni.sig(ans, model="log-add", alpha=0.05,
+   include.all.SNPs=FALSE)
number of tests:  21
alpha: 0.05
corrected alpha: 0.002380952
          comments log-additive
snp10001  -        0.001143723
snp100024 -        0.002231790
```

In this case we have corrected using the number of test performed only in those SNPs that are not monomorphic and in which the percentage of genotyping is greater than those established in `WGassocation` function. The argument `include.all.SNPs` may be modified in order to include all SNPs analyzed.

3.2.2 FDR correction

False Discovery Rate (FDR) is an approach to the multiple comparisons problem. Instead of controlling the chance of any false positives (as Bonferroni do), FDR controls the expected proportion of false positives [Benjamini and Hochberg, 1995]. The `R` library `qvalue` performs the FDR analysis. The only think it needs is a vector with the pvalues. These may be easily obtained after executing `WGassociation` using the functions `codominant, dominant, recessive, overdominant or additive` depending on the model we are interested in analyzing. As an example, let us assume that we want to compute the FDR for the SNPs in the HapMap example. The p values are saved in the object `resHapMap`, so they may be obtained using:

```
>pvalAdd<-additive(resHapMap)
```

Notice that `codominant(resHapMap)` would not work since we only computed the log-additive model for these data. Now we have to delete those SNPs that are monomorphic. This can be done executing:

```
>pval<-pval[!is.na(pval)]
```

After that, the q-values can be calculated as follows:

```
>library(qvalue)
>qobj<-qvalue(pval)
```

Finally, if we are interested in knowing the FDR for a desired p-value (e.g. 0.001) we might try:

```
> max(qobj$qvalues[qobj$pvalues <= 0.001])
[1] 0.0005786454
```

Other methods based on p values as implemented in `R` package `multtest` [Pollard et al, 2006] could be used. As an example, we could use the following testing procedures based on permutation adjusted p-values:

```
procs<-c("Bonferroni","Holm","Hochberg","SidakSS","SidakSD","BH","BY")
res2<-mt.rawp2adjp(rawp,procs)
```

Then, the identity and number of rejected hypotheses for previous multiple testing procedures and different nominal Type I error rates may easily obtained typing:

```
 mt.reject(cbind(res$rawp,res$adjp),seq(0,0.1,0.001))$r
      rawp Bonferroni Holm Hochberg SidakSS SidakSD   BH   BY
0        0          0    0        0     220     220    0    0
0.001 3343       1519 1538     1538    1519    1538 3100 2454
0.002 3550       1592 1651     1651    1595    1651 3324 2643
0.003 3732       1672 1706     1706    1672    1706 3488 2781
0.004 3786       1712 1783     1783    1713    1783 3543 2830
0.005 3846       1752 1812     1812    1752    1813 3612 2876
0.006 3894       1801 1832     1832    1801    1833 3736 2927
0.007 4011       1818 1856     1856    1818    1856 3765 3036
0.008 4047       1832 1874     1874    1833    1875 3802 3052
0.009 4081       1851 1888     1888    1852    1890 3833 3087
0.01  4122       1867 1931     1931    1868    1931 3855 3112
....
```

4 Analysis of multiple SNPs

4.1 Haplotype analysis

Library `haplo.stats` is specifically designed to deal with haplotype estimates. The `haplo.glm` function performs a regression of a given trait (quantitative or not) on ambiguous haplotypes using a general lineal model (glm). As the authors point out, the "critical" element of the data frame to fit these models is the matrix of genotypes. Thus we have programmed a function called `make.geno` that prepares the SNPs in the required format to be included in the formula of `haplo.glm` function. A regression analysis may then be performed as follows. Let us assume that we know that the tag SNPs are `snp10001, snp100019` and `snp100029`. We first prepare a model matrix with these genotypes to be analyzed in `haplo.glm` function as follows:

```
> datSNP<-setupSNP(SNPs,6:40,sep="")
> tag.SNPs<-c("snp100019", "snp10001", "snp100029")
> geno<-make.geno(datSNP,tag.SNPs)
```

We must notice that the order of the SNPs is important and this is why we have written `tag.SNPs<-c("snp100019", "snp10001", "snp100029")`. This information must be known by the user. After that, we can easily estimate the effects of haplotypes using `haplo.glm` function as follows:

```
> mod<-haplo.glm(log(protein)~geno,data=SNPs,
+ family=gaussian,
+ locus.label=tag.SNPs,
+ allele.lev=attributes(geno)$unique.alleles,
+ control = haplo.glm.control(haplo.freq.min=0.05))
> mod

  Call:
haplo.glm(formula = log(protein) ~ geno,
    family = gaussian, data = SNPs, locus.label = tag.SNPs,
    allele.lev = attributes(geno)$unique.alleles,
    control = haplo.glm.control(haplo.freq.min = 0.05))

Coefficients:
               coef     se  t.stat     pval
```

```
(Intercept) 10.6880 0.0985 108.543 0.00e+00
geno.3      -0.3485 0.0859  -4.058 7.86e-05
geno.6      -0.0466 0.0994  -0.469 6.40e-01
geno.rare   -0.2324 0.2429  -0.957 3.40e-01

Haplotypes:
           snp100019 snp10001 snp100029 hap.freq
geno.3             G        C         A   0.2321
geno.6             G        T         G   0.2990
geno.rare          *        *         *   0.0262
haplo.base         C        T         G   0.4427
```

The method `intervals` have been designed to obtain the confidence intervals for an object of class `haplo.glm`

```
> intervals(mod)
               freq   diff   95%   C.I.         P-val
         CTG 0.4351  10.70 Reference haplotype
         GCA 0.2366  -0.36 (  -0.53 -  -0.19 )  0.0000
         GTG 0.3016  -0.05 (  -0.25 -   0.15 )  0.6112
        rare 0.0267  -0.24 (  -0.72 -   0.24 )  0.3219
```

Other covariates may also be included in the model (adjusted analysis) as usual (e.g. `log(protein)  geno + sex`). When case-control study is performed, we need to change `family=gaussian` by `family=binomial` when `in haplo.glm` is executed.

We can also perform and interaction analysis between haplotypes and a factor variable. The function `haplo.interaction` calls both `make.geno` and `haplo.glm` functions to perform the following analysis:

```
     Haplotype using SNPs: snp100019 snp10001 snp100029  adjusted by:
 Interaction
-------------------------
     HapFreq Male (dif) lower upper Female (dif) lower upper
CTG   0.4427       0.00    NA    NA        -0.20 -0.58  0.18
GTG   0.2983      -0.16 -0.46  0.13        -0.10 -0.43  0.23
GCA   0.2313      -0.30 -0.53 -0.07        -0.62 -1.00 -0.25
rare  0.0278       0.00 -0.60  0.59        -0.96 -1.82 -0.10

p interaction: 0.1361707

 sex within haplotype
-------------------------
$Male
      diff lower upper
CTG   0.00    NA    NA
GTG  -0.16 -0.46  0.13
GCA  -0.30 -0.53 -0.07
rare  0.00 -0.60  0.59

$Female
      diff lower upper
CTG   0.00    NA    NA
GTG   0.09 -0.17  0.35
GCA  -0.42 -0.68 -0.17
rare -0.76 -1.58  0.05
```

```
haplotype within sex
-------------------------
$CTG
       diff lower upper
Male    0.0    NA    NA
Female -0.2 -0.58  0.18

$GTG
       diff lower upper
Male   0.00    NA    NA
Female 0.06 -0.22  0.34

$GCA
        diff lower upper
Male    0.00    NA    NA
Female -0.32 -0.66  0.01

$rare
        diff lower upper
Male    0.00    NA    NA
Female -0.96 -1.96  0.04
```

4.2 Gene-Gene interaction analysis

The last analysis we can perform with `SNPassoc` package is an interaction analysis between SNPs. This analysis makes sense in the case of having SNPs from different genes or chromosomes. This analysis may be done using `interactionPval` function. The command is:

```
> ansCod<-interactionPval(log(protein)~sex, data=myData.o,
+          model="codominant")
```

The meaning of this instruction is the following. We are looking for interactions effects of protein levels (in log scale) between the SNPs: `snp10001, snp10002, ... , snp100035` adjusted by sex. This function requires that a model of inheritance is specified. In this case we assume a codominant model. The `ansCod` matrix may be printed. The upper part of the matrix contains the p values for the interaction (epistasis) log-likelihood ratio (LRT) test. The diagonal contains the p values from LRT for the crude effect of each SNP. Finally, the lower triangle contains the p values from LRT comparing the two-SNP additive likelihood to the best of the single-SNP models. This information may also be plotted using `plot` function obtaining the plots showed in Figure 8.

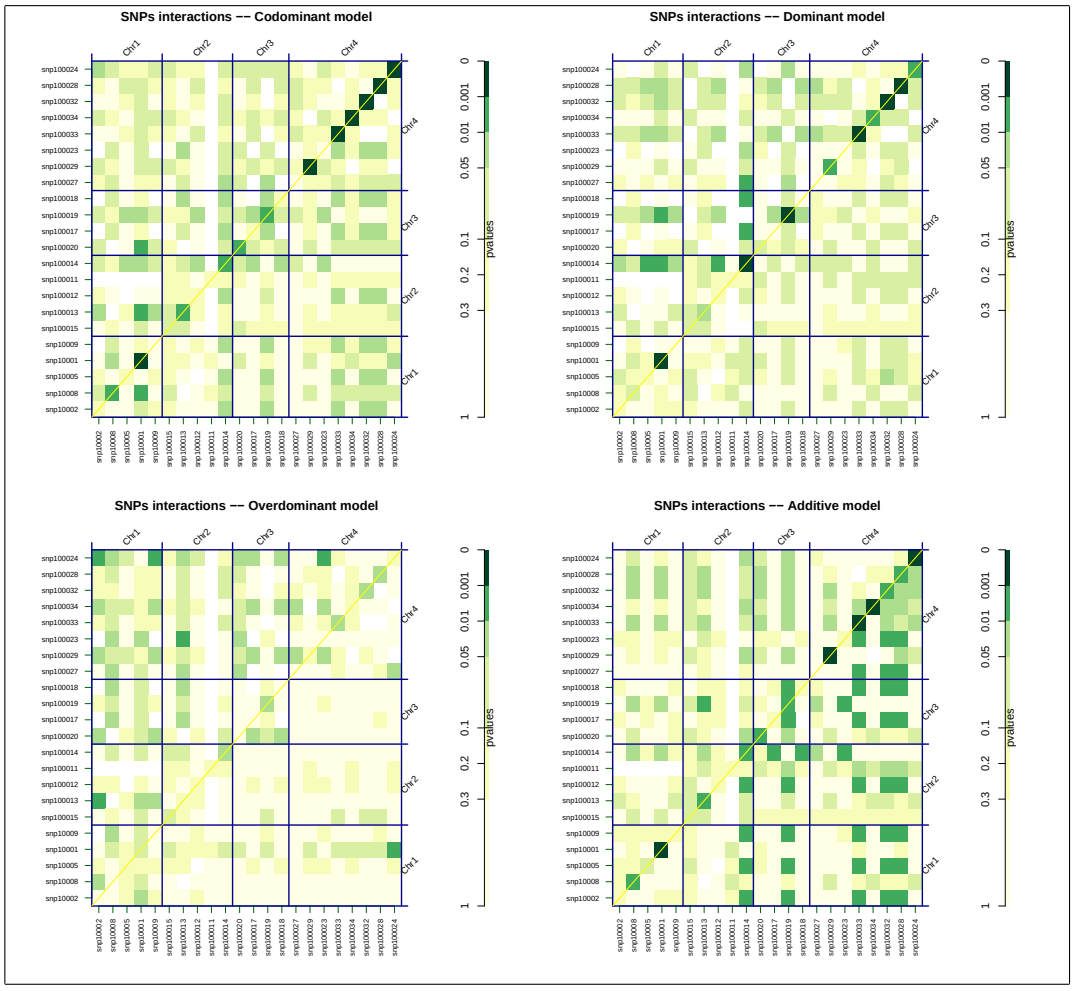

Figure 8: Interaction plots for each SNP using four genetic models. Each plot contains the p values obtained from different likelihood ratio tests. Different colors indicates different statistical significant levels. The diagonal contains the p values from likelihood ratio test for the crude effect of each SNP. The upper triangle in matrix contains the p values for the interaction (epistasis) log-likelihood ratio test. Finally, the lower triangle contains the p values from LRT comparing the two-SNP additive likelihood to the best of the single-SNP models.

5 Statistical Methods

5.1 Association between a single SNP and a trait

To study the association between a given SNP and a trait (function `association`, we may consider a SNP as a categorical variable with one level for each possible genotype (codominant model). In such a situation, to assess the association between the phenotype Y (quantitative or binary) and a SNP, we apply a general linear model (glm):

$$Y_i = \alpha + \beta X_i + \epsilon_i, \tag{1}$$

where α is the intercept, X_i is the i^{th} subject's genotype score for a given marker and ϵ_i is distributed according to a normal with mean 0 and variance σ^2. Under the additive model, X_i indicates i^{th} subjects' number of minor alleles; under the dominant model, X_i denotes, with coded values 1 and 0, whether the i^{th} subject has at least one minor allele. Similarly, under the recessive (or over-dominant) model, X_i is codified as 1 and 0 depending on whether the i^{th} subject has two minor alleles (or, in the over-dominant model, two minor or two major alleles). Depending on Y's distribution (normal or binomial) the most appropriate link function may be chosen. Confounded association is an important point in genetic association studies [Cordell and Clayton, 2005]. In case we need to adjust the model by confounders' variables, the equation (1) may be easily extended just adding the term, γZ_i, where Z denotes confounders' variables. For each genetic model we compute the odds ratios, $\exp(\beta)$, for dichotomous traits and mean differences for quantitative traits. Confidence intervals are also computed using the variance estimated for each parameter.

5.2 Genetic model selection

To test the statistical significance of a given SNP, we compare the effect of the polymorphism with the null model (only including the intercept) using the likelihood ratio test, $\text{LRT} = 2(\log \text{Lik}_{null} - \log \text{Lik}_{other})$, where "other" makes reference to codominant, recessive, dominant, overdominant or additive. In some occasions, when case-control studies are analyzed, this test cannot be applied since there are no cases in a given cell. In that case `association` functions compute the exact Fisher test instead of LRT. These p values are showed in the output after using the functions: `association`, `WGassociation`, `scanWGassociation`.

When this test is not sensitive enough to discriminate between models, other criteria, like the Akaike information (AIC), may be useful to choose the right model of inheritance. In general, the most optimal is attributed to the model with the less AIC, $\text{AIC} = -2 \log \text{Lik} + 2q$, where q denotes the number of parameters for the fitted model. The AIC is given in the last column in the output for `association` function.

5.3 Analysis of multiple SNPs

5.3.1 Interaction between SNPs

The study of more than one SNP at the same time, and their interactions, may be easily introduced in Equation 1. The `interactionPval` fuction calculates, for each pair of SNPs (i,j): the likelihood underlying the null model Lik_{null} (e.g. only with α), the likelihood under each of the single-SNP, Lik_i and Lik_j, the likelihood under an additive SNP model $\text{Lik}_{ad(i,j)}$, and the likelihood under a full SNP model (including SNP-SNP interaction), $\text{Lik}_{full(i,j)}$. Here `SNPassoc` uses the object-oriented features of `R` ("classes and methods") to plot interaction analysis. If `ans` is an object of class `SNPinteraction` then `plot(ans)` will generate a plot with the following information. The upper triangle in matrix from this function contains the p values for the interaction (epistasis) log-likelihood ratio test, LRT,

$\text{LRT}_{ij} = -2(\log \text{Lik}_{full(i,j)} - \log \text{Lik}_{ad(i,j)})$. The diagonal contains the p values from LRT for the crude effect of each SNP, $\text{LRT}_{ii} = -2(\log \text{Lik}_i - \log \text{Lik}_{null})$. The lower triangle contains the p values from LRT comparing the two-SNP additive likelihood to the best of the single-SNP models, $\text{LRT}_{ji} = -2(\log \text{Lik}_{ad(,j)} - \log \max(\text{Lik}_i, \text{Lik}_j))$.

5.3.2 Haplotype analysis

Haplotype analysis is performed calling functions from the `haplo.stats` package [Sinnwell and Schaid, 2005] which implements the EM algorithm. The authors proposed a method to study haplotype association that are applicable to either dichotomous or quantitative traits using generalized linear models (see for further details [Schaid et al, 2002]). Our contribution has been to program several functions and methods (i.e., `make.geno` or `intervals`) to deal with genotype matrices (required by `haplo.glm` function). In addition, as in the case of SNPs, we may be interested in assessing the effect of gene-environment interaction. Lake et al, 2003 extended the method of Schaid et al, 2002 to tests and estimation of haplotype-environment interaction. The association analysis of haplotypes is similar to that above described of genotypes in that either logistic regression are shown as OR and 95% CI or linear regression results with differences in mean effects and 95% CI (function `haplo.interaction`).

6 Computational issues

It is well known that association studies at a whole genome scale are a very time consuming task due to the large amount of SNPs that are analyzed. Besides the statistical procedures, there are also other steps we have to perform before further analyzing the data. The first step is to import SNP data. This information is usually available in a text file. The simplest way (most user-friendly) of importing such kind of data to `R` is using either `read.table` or `read.delim` functions. The problem with using these functions is their computational cost. Thus, we strongly recommend the user to employ `scan` function which is very much less time demanding. Here you may see a possible way of importing genotype data to `R`. Let us assume that `HapMap.txt` contains the genotype data and that the first row has the names of SNPs.

```
n<-120 #number of rows without the header (e.g. number of individuals)

dat<-scan("HapMap.txt",list("character"),skip=1)

variables<-scan("HapMap2.txt",list("character"),n=1)

ncols<-length(dat[[1]])/n

temp<-matrix(dat[[1]],nrow=n,ncol=ncols,byrow=TRUE)

HapMap<-data.frame(temp, stringsAsFactors = FALSE)

dimnames(HapMap)[[2]]<-variables[[1]]
```

The second step is to prepare the data for being analyzed. That is, we need to indicate which variables are SNPs. To do this, `setupSNP` function is used. Lastly, the statistical test is carried out using either `WGassociation` or `scanWGassociation` functions. As it has been indicated in this manual, the first one gives a summary of association tests (sample sizes, ORs or mean differences, confidence intervals, likelihood ratio tests, and AIC), while the second one is focused on computing the p values corresponding to the likelihood ratio test. Table 2 shows an estimated time cost of these procedures. As you may observe a whole genome association study including close to 270,000 SNPs may be carried out in approximately 1 hour. We must indicate that associations are performed only in 10 minutes.

Study	n	SNPs	action	R function	CPU time 1 model / 5 models
Quantitative trait	150	35	import data	read.table	0.1sec
			prepare data	setupSNP	0.2sec
			summary	WGassociation	1.1sec / 3.6sec
			compute p values	scanWGassociation	0.2sec / 1.0sec
Case/control	110/47	35	summary	WGassociation	1.2sec / 3.8sec
			compute p values	scanWGassociation	0.3sec / 1.4sec
			interaction	interactionPval	23sec / 1min 50sec
Case/control	369/341	138	import data	read.table	0.3sec
			prepare data	setupSNP	1.2sec
			summary	WGassociation	2.6sec / 7.4sec
			compute p values	scanWGassociation	0.6sec / 3.1sec
			interaction	interactionPval	8min 20sec / 32min 30sec
Two groups	60/60	9,305	import data	scan	2min 10sec
(HapMap)			prepare data	setupSNP	1min 15sec
			summary	WGassociation	13min 15sec / 1h 09min
			compute p values	scanWGassociation	1.9sec / 3.2sec
			permutation test	scanWGassociation	4min 50sec / -
Two groups	60/60	269,605	import data	scan	6min 32sec
(HapMap)			prepare data	setupSNP	58min 13sec
			summary	WGassociation	Not calculated
			compute p values	scanWGassociation	1min 42sec / 3min 38sec
			permutation test	scanWGassociation	3h 20min / -

Table 2: CPU time requirements for different procedures involved in a whole genome analysis. The analysis was carried out in a dual AMD-64 bit processor. The action summary includes to compute odds ratios, their confidence intervals, p values and AIC.

7 Acknowledgments

Authors thank Mònica Gratacòs, and Rafael de Cid for testing the functions and feedback that led to improve the package. Mario Cáceres for critical reading and helpful comments about the manuscript. Frank Dudbridge is also acknowledge for his comments about permutation test and for sending us some `R` code to perform rank truncated product test. We are also grateful to Fabio Sánchez, Ji Young Lee, David Cairns, Daryl Waggott, and Francesc Castro for validating the functions using their own data.

This work has been supported by CEGEN (Spanish Genotyping Center) funded by Genoma España and grant FIS03/0144 (Instituto de Salud Carlos III). JMM was supported by the CRG under project SAF2005-01005 (Spanish Ministry of Science and Education) and the Danone Institute.

References

[Benjamini and Hochberg, 1995] Benjamini, Y., and Hochberg Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing". *Journal of the Royal Statistical Society. Series B*, **57**, 289300.

[Dudbridge et al, 2006] Dudbridge, F., Gusnanto, A. and Koeleman, B.P.C. (2006) "Detecting multiple associations in genome-wide studies". *Human Genomics*, **2**, 310-317.

[Dudbridge and Koeleman, 2004] Dudbridge F and Koeleman BPC (2004) "Efficient computation of significance levels for multiple associations in large studies of correlated data, including genomewide association studies". *Am J Hum Genet*, **75**, 424-435.

[Cordell and Clayton, 2005] Cordell, H. J. and Clayton, D. G. (2005) Genetic association studies, *Lancet*, **366**, 1121-1131.

[Lake et al, 2003] Lake, S. L. and Lyon, H. and Tantisira, K. and Silverman, E. K. and Weiss, S. T. and Laird, N. M. and Schaid, D. J. (2003), Estimation and tests of haplotype-environment interaction when linkage phase is ambiguous, *Hum Hered*, **55**, 56-65.

[Pollard et al, 2006] Pollard,K.S., Ge,Y. and Dudoit,S. (2006) multtest:Resampling-based multiple hypothesis testing. R package version 1.10.2 http://cran.r-project.org.

[Sinnwell and Schaid, 2005] Sinnwell, J. P. and Schaid, D. J. (2005) haplo.stats: Statistical analysis of haplotypes with traits and covariates when linkage phase is ambiguous. R package version 1.2.2. http://mayoresearch.mayo.edu/mayo/research/biostat/schaid.cfm.

[Schaid et al, 2002] Schaid, D. J. and Rowland, C. M. and Tines, D. E. and Jacobson, R. M. and Poland, G. A. (2002), Score tests for association between traits and haplotypes when linkage phase is ambiguous, *Am J Hum Genet*, **70**, 425-434.