GenABEL tutorial

Yurii Aulchenko

February 21, 2008

Contents

1 Overview

GenABEL is an R library developed to facilitate Genome-Wide Association (GWA) analysis of binary and quantitative traits. GenABEL is implemented as an R library. R is a free, open source language and environment for general-purpose statistical analysis (available at http://www.r-project.org/). It implements powerful data management and analysis tools. Though it is not strictly necessary to learn everything about R to run GenABEL, it is highly recommended as this knowledge will improve flexibility and quality of your analysis.

This tutorial was originally written to serve as a set of exercises for the "Advances in population-based studies of complex genetic disorders" (GE03) course of the Netherlands Institute of Health Sciences (Nihes).

If you read this tutorial not as a part of the GE03 course, and you are eager to start with you GWA analysis without reading all the not-so-strictly-necessary staff, start directly from the section 5 ("Genome-wide association analysis").

Otherwise, you can start with R basics and simple association analyises using few SNPs in section 2, "Introduction to R". In the next section, 4 ("Introduction to `GenABEL` ") you will learn how to work with the `gwaa.data-class`, which is used to store GWA data in `GenABEL` and will perform some simple large-scale analyses.

In the next section, 5 ("Genome-wide association analysis"), you will do quality control of genetic data and do association analysis under realistic conditions. This section is the core of this tutorial.

The section 6 ("GWA in presence of genetic stratification") is dedicated to analysis in presence of populational stratification and analysis of family-based data.

Genetic data imputations are covered in section 8, "Genetic data imputations".

The last section, 10 ("Analysis of selected region"), is dedicated to analysis of haplotype association and analysis of SNP interactions.

Information on importing the data from different formats to `GenABEL` is given in appendix A. Answers to exercises are provided in appendix B.

Experienced R users start directly with the section (4, "Introduction to `GenABEL` ").

2 Introduction to R

In this section we will consider basic R data types and operations, tools for analysis of qualitative and quantitative traits. Needless to say, that only basic R functionality may be covered within such short section. If you decide to improve your knowledge of R, we recommend you excellent manuals, such as 'An introduction to R', 'Simple R', 'Practical Regression and Anova using R', and others, available free of charge from the R project web-site (http://www.r-project.org).

In the first parts of this section, we will cover basic types of R objects and will work with them. Next, we will analyse qualitative and quantitative traits, using only basic R functionality.

2.1 Basic R data types and operations

On the contrast to many other statistical analysis package, analysis in R is not based on graphic user interface, but is rather command-based. When you first start R, a command prompt appears. To get help and overview of R, type `help.start()` in the command line and press `enter`. This will start your internet browser and open R documentation.

Let us first exploit R as a calculator. You can directly operate with numbers in R. Try adding three to two:

```
> 2 + 3
```

```
[1] 5
```

Square roots, base-10 logarithm, and exponentiation can be done straightforwardly with

```
> sqrt(5)

[1] 2.236068

> log10(2.24)

[1] 0.350248

> exp(0.35)

[1] 1.419068
```

The arithmetic operations and functions can be nested:

```
> exp(log10(sqrt(2 + 3)))

[1] 1.418337
```

R functions include standard ones, such as logarithms, power, but also a wide range of statistical function, for example, distribution and probability density of many distributions.

For any function with name say '`fun`', help may be obtained by typing '`help(fun)`' on the command line. If you do not know the exact name for the function you look for, try '`help.search("query")`', where `query` is the keyword.

Exercise 1 *Try to find out what are the functions to do*

1. *Wilcoxon test*
2. *Fisher exact test*
3. *T-test*

R help pages have standard layout, documenting usage of the function, explaining function arguments, providing details of implementation, explaining the value returned by the function, and giving references and examples of function usage.

Exercise 2 *Explore the help page for the Wilcoxon test and answer the questions:*

1. *When exact Wilcoxon test is computed by default?*
2. *If the default conditions for the exact test are not satisfied, what approximation is used?*

Most of the documented functions will have examples of their usage at the end of the 'help' page, and these examples can be evaluated in R. Try '`example(wilcox.test)`'.

One of important R operations is *assignment*, which is done with '`<-`' operator. For example, we if want to assign value '2' to variable '`a`', and value '3' to the variable '`b`':

```
> a <- 2
> b <- 3
```

Typing the variable name in R command line will return its' value, e.g.

```
> b

[1] 3
```

Naturally, evaluation of

```
> exp(log10(sqrt(a + b)))

[1] 1.418337
```

gives the expected result we have obtained above using numerical arguments.

Just introduced variables 'a' and 'b' contain single values. More practically important are the variables containg *vectors*. Let us create an example vector and experiment with it:

```
> v <- c(1, 3, 5, 7, 11)
```

Now, let us try different operations with this vector:

```
> v + 1

[1]  2  4  6  8 12

> 1/v

[1] 1.0000000 0.3333333 0.2000000 0.1428571 0.0909091

> log(v)

[1] 0.000000 1.098612 1.609438 1.945910 2.397895
```

It is easy to see that these operations give a vector with elements being the result of application of the same operation to different initial elements.

What happens if two vectors are supplied as function arguments? Let us define new vector

```
> ov <- c(1, 2, 3, 4, 5)
```

and add it and previous vector up:

```
> v + ov

[1]  2  5  8 11 16
```

multiply them

```
> v * ov

[1]  1  6 15 28 55
```

or compute `v` to the power of `ov`:

```
> v^ov

[1]      1      9    125   2401 161051
```

Here, the operation was done over each of the two corresponding elements of the vectors, resulting, again, in the vector of the same length.

Other functions may evaluate a vector as a whole an return a single value as output. For example, to obtain a sum of vector's elements, use

```
> sum(v)

[1] 27
```

Other examples of such functions involve `length`, returning number of elements of a vector, `mean`, returning the mean, `var`, returning the variance, etc. For example:

```
> length(v)

[1] 5

> mean(v)

[1] 5.4

> var(v)

[1] 14.8
```

One of the most important data operations in R is *sub-setting*. This refers to operations which help you deriving a subset of the data. Let us create a short vector and play a bit with sub-setting. This vector will contain 5 simple character strings:

```
> a <- c("I am element 1", "I am element 2", "I am element 3",
+     "I am element 4", "I am element 5")
> a

[1] "I am element 1" "I am element 2" "I am element 3" "I am element 4"
[5] "I am element 5"
```

To find out what is the value of the i-th element of this vector, you can sub-set it by `a[i]`. For example the 3rd elements is:

```
> a[3]

[1] "I am element 3"
```

You can also select bigger sub-sets, e.g. all elements from 2 to 4:

```
> a[c(2:4)]

[1] "I am element 2" "I am element 3" "I am element 4"
```

Here, operation `c(2:4)` is equivalent to 'sequence from 2 to 4', that is `c(2,3,4)`.

We can easily get disjoint elements; e.g. if you want to retrieve elements 1, 3, and 5, you can do

```
> dje <- c(1, 3, 5)
> dje

[1] 1 3 5

> a[dje]

[1] "I am element 1" "I am element 3" "I am element 5"
```

One of very attractive features of R data objects is possibility to derive sub-sets based on some condition. Let us consider two vectors, `tmphgt`, containing the height of some subjects, and `tmpids`, containing their IDs:

```
> tmphgt <- c(150, 175, 182, 173, 192, 168)
> tmphgt

[1] 150 175 182 173 192 168

> tmpids <- c("fem1", "fem2", "man1", "fem3", "man2", "man3")
> tmpids

[1] "fem1" "fem2" "man1" "fem3" "man2" "man3"
```

Imagine you want to derive the IDs of people with height over 170 cm. The way to do it is to combine several steps. First, you can run the logical function `>170` on the height data:

```
> vec <- (tmphgt > 170)
> vec

[1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE
```

This returns logical vector whose elements are true, when particular element of `tmphgt` satisfies the condition `>170`. Such logical vector, in turn, may be applied to sub-set any other vector of the same length[1], including itself. If you want to see what are the heights in people taller than 170 cm, you can use

```
> tmphgt[vec]

[1] 175 182 173 192
```

or you can get IDs of these people with

[1] Actually, you can apply it to a longer vector too, and then the logical vector will be "expanded" to total length by repeating the original vector head-to-tail. However, we will not use this in our exercises.

```
> tmpids[vec]

[1] "fem2" "man1" "fem3" "man2"
```

You can combine more than one logical condition to derive sub-sets. For example, to see what are the IDs of people taller than 170 but shorter than 190 cm, you ca use

```
> vec <- (tmphgt > 170 & tmphgt < 190)
> vec

[1] FALSE  TRUE  TRUE  TRUE FALSE FALSE

> tmpids[vec]

[1] "fem2" "man1" "fem3"
```

Other, and easier[2] way to do the same is to use `which()` function. This function reports which elements pass logical condition. To obtain above results you can run:

```
> vec <- which(tmphgt > 170 & tmphgt < 190)
> vec

[1] 2 3 4

> tmpids[vec]

[1] "fem2" "man1" "fem3"
```

You can remove `tmphgt` and `tmpids` variable so they will not interfere with our future analysis:

```
> ls()

[1] "a"      "b"      "dje"    "ov"     "tmphgt" "tmpids" "v"      "vec"

> rm(tmphgt, tmpids)
```

and check if the removal was successful:

```
> ls()

[1] "a"   "b"   "dje" "ov"  "v"   "vec"
```

Sub-setting for 2D objects (matrices) is done in similar manner. Let us construct a simple matrix and do several sub-setting operations on it:

```
> a <- matrix(c(11, 12, 13, 21, 22, 23, 31, 32, 33), nrow = 3,
+     ncol = 3)
> a
```

[2] Because it treats NAs for you

```
     [,1] [,2] [,3]
[1,]   11   21   31
[2,]   12   22   32
[3,]   13   23   33
```

To obtain the element in the 2nd row and 2nd column

```
> a[2, 2]

[1] 22
```

...the second row and third column:

```
> a[2, 3]

[1] 32
```

To obtain the 2x2 set of elements contained in upper left corner, you can do

```
> a[1:2, 1:2]

     [,1] [,2]
[1,]   11   21
[2,]   12   22
```

Or you can even get the variables, which reside in corners:

```
> a[c(1, 3), c(1, 3)]

     [,1] [,2]
[1,]   11   31
[2,]   13   33
```

If one of the dimensions is not specified, complete list is returned for this dimension. For example, here we retrieve the first row

```
> a[1, ]

[1] 11 21 31
```

...and third column

```
> a[, 3]

[1] 31 32 33
```

...or columns 1 and 3:

```
> a[, c(1, 3)]

     [,1] [,2]
[1,]   11   31
[2,]   12   32
[3,]   13   33
```

As well as with vectors, you can sub-set matrices using logical conditions or indexes. GWA genetic data are stored in matrices, and you can sub-set them using the methods described above.

For example, if we want to see what elements of a are greater than 21, we can run

```
> a > 21
```

```
      [,1]  [,2] [,3]
[1,] FALSE FALSE TRUE
[2,] FALSE  TRUE TRUE
[3,] FALSE  TRUE TRUE
```

or obtain these elements by

```
> a[a > 21]
```

```
[1] 22 23 31 32 33
```

At this point, you can exit R.

Summary:

- You can get a top-level access to R documentation by `help.start()`. To search help for some keyword `key`, try `help.search(key)`. To get description of some function `fun`, try `help(fun)`.
- You can use R as a powerful calculator.
- It is possible to get sub-sets of vectors and matrices by specifying index value or a logical condition (of the same length as the vector / matrix) between square brackets (`[, ]`)
- When you obtain an element of a matrix with `[i,j]`, `i` is the row and `j` is the column of the matrix.
- Function `which(A)` returns index of the elements of A which are "true".

Exercise 3 *In this exercise, you will explore few vectors representing different data on study subjects described in* `srdta` *example data set supplied together with* `GenABEL` *. First, you need to load* `GenABEL` *by typing*

```
> library(GenABEL)
```

and load the data by

```
> data(srdta)
```

The vector containing study subjects sex can be accessed through `srdta@gtdata@male`*; this vector's value is one when the corresponding person is male and zero otherwise. The vector containing SNP names can be accessed via* `srdta@gtdata@snpnames`*, chromosome ID – through* `srdta@gtdata@chromosome` *and map – through* `srdta@gtdata@map`*. Explore these vectors and answer the questions.*

1. *What is the ID and sex of the first person in the data set?*
2. *Of the 22nd person?*
3. *How many males are observed among first hundred subjects?*
4. *How many FEMALES are among 4th hundred?*
5. *What is the male proportion in first 1000 people?*
6. *What is the FEMALE proportion in second 1000 (1001:2000) people?*
7. *What is name, chromosome and map position of 33rd maker?*
8. *What is distance between markers 25 and 26?*

2.2 Data frames

Start R with double-click on the file named `assocbase.RData`. You can see the names of the loaded objects by using the command `ls()`:

```
> ls()

[1] "assoc"
```

You can see that there is a single object the class of the object can be interrogated by using `class` function:

```
> class(assoc)

[1] "data.frame"
```

Thus, the file you have loaded contains one *data frame*. A data frame is an `R` term for a data table. In such tables, it is usually assumed that rows correspond to subjects (observations) and columns correspond to variables.

We will investigate the data presented in the `assoc` data frame. The nice feature of data frames is that columns carry names for the variables, and the data stored there can be retrieved by referencing these names. [3] To see what are the variable names, use the command `names()`:

```
> names(assoc)
```

[3] This may also be true for matrices; more fundamental difference is though that a matrix always contains variables of the dame type, e.g. character or integer, while a data frame may contain variables of different types.

```
[1] "subj" "sex"  "aff"  "qt"   "snp4" "snp5" "snp6"
```

The 7 variables correspond to the personal ID, sex, affection status, quantitative trait `qt` and several SNPs.

A variable from a data frame `frame`, which has some name `name` can be accessed through `frame$name`. This will return a conventional vector.For example to see the affection status (`aff`) in the data frame `assoc`, use

```
> assoc$aff
```

```
  [1] 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0
 [38] 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0
 [75] 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0
[112] 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1
[149] 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1
[186] 1 1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0
[223] 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0
```

Exercise 4 *Investigate types of the variables presented in data frame* `assoc`*. For each variable, write down the class.*

We can easily check how many people are described in the data set by checking the length of any variable (all variables will be the same length in a data frame):

```
> length(assoc$sex)
```

```
[1] 250
```

or through asking to report the dimensionality of the data frame:

```
> dim(assoc)
```

```
[1] 250   7
```

Here, the first number corresponds to the number of rows (subjects) and the second to the number of columns (variables).

Let us go into more details and check what are the ID, sex and affection status for the person 75:

```
> assoc$subj[75]
```

```
[1] 1409
```

```
> assoc$sex[75]
```

```
[1] 1
```

```
> assoc$aff[75]
```

```
[1] 0
```

Data frame may be thought of as a matrix, or as a collection of vectors. All sub-setting operations discussed before for vectors/matrices are applicable to a data frame. Therefore an alternative way to see all the data for person 75 is:

```
> assoc[75, ]
```

```
     subj sex aff       qt snp4 snp5 snp6
1409 1409   1   0 1.014664  A/B  B/A  B/B
```

In the same manner, you can get data for e.g. subjects 5 to 15 (a new data frame containing data only on first 10 people) by

```
> assoc[5:15, ]
```

```
     subj sex aff         qt snp4 snp5 snp6
1533 1533   0   0  0.1009220  A/B  B/A  B/A
2466 2466   1   0 -0.1724321  A/B  A/A  A/A
2425 2425   0   0 -0.3378473  B/B  A/A  A/A
1068 1068   0   0 -1.7112925  A/A  B/B <NA>
198   198   1   0 -0.4815822  A/B  B/A  B/A
1496 1496   1   0  1.2281232  A/A  B/B  B/B
909   909   0   0  0.5993945  A/B  B/A  B/A
1213 1213   0   0  1.9792190  A/A  B/B  B/B
181   181   1   0  1.5435921  A/A  B/B  B/B
1783 1783   0   0 -1.6242738  A/B  B/A  B/A
1914 1914   0   0 -0.5160331  A/A  B/B  B/B
```

As well as with vectors, it is possible to sub-set elements of a data frame based on (a combination of) logical conditions. For example, if we want to check what are the IDs of the people with `qt` over 1.4, we can run

```
> vec <- which(assoc$qt > 1.4)
> vec
```

```
 [1]  12  13  33  41  54  68  72  76  89 106 118 142 156 161 175 181 193 219 241
```

```
> assoc$subj[vec]
```

```
 [1] 1213  181 1737 1319  516 1355  186 1426 1284  822 2129  212 1443  704 1648
[16] 1628  562  858  698
```

Or, if we are interested to see subject-number and SNP information for these people, we can see that with

```
> assoc[vec, c(1, 5, 6, 7)]
```

```
     subj snp4 snp5 snp6
1213 1213  A/A  B/B  B/B
181   181  A/A  B/B  B/B
1737 1737  A/A  B/B  B/B
1319 1319  A/A  B/A  B/A
```

```
516   516  A/B  B/A  B/A
1355 1355  A/A  B/B  B/B
186   186  A/A  B/A  B/A
1426 1426  A/B  B/A  B/A
1284 1284  A/A  B/B  B/B
822   822  A/B  B/A  B/A
2129 2129  A/B  B/A  B/A
212   212  A/B  B/A  B/A
1443 1443  A/A  B/B  B/B
704   704  A/B  B/A  B/A
1648 1648  A/B  B/A  B/A
1628 1628  A/B  B/A  B/A
562   562  A/A  B/B  B/B
858   858  A/B  B/A  B/A
698   698  B/B  A/A  A/A
```

here, we select people identified by `vec` in the first dimension (subjects), and by `c(1,5,6,7)` we select first, fifth, sixth and seventh column (variable).

Next, we can check how many of the subjects are males by

```
> sum(assoc$sex)

[1] 129
```

and what is male sex proportion:

```
> sum(assoc$sex)/length(assoc$sex)

[1] 0.516
```

or

```
> mean(assoc$sex)

[1] 0.516
```

or

```
> sum(assoc$sex == 1)/length(assoc$sex)

[1] 0.516
```

You can also explore and modify the raw data contained in a data frame by using `fix()` command (e.g. `fix(assoc)`). However, normally this is not necessary.

The function `table(x)` produces a frequency table for the variable `x`. Thus, we can use

```
> table(assoc$sex)

  0   1
121 129
```

which, again, tells us that there are 129 males and 121 females in this data set.

Tables of other qualitative variables, such as affection and SNPs, can be generated in the same manner.

Exercise 5 *Explore qualitative variables presented in* `assoc`

1. *How many affected and unaffected are present in the data set?*
2. *What is the proportion of affected?*
3. *What is the distribution of snp4?*

A more convenient way to access data presented in a data frame is through "attaching" it to the R search path by

```
> attach(assoc)
```

After that, the variables can be accessed directly, e.g.

```
> table(sex)

sex
  0   1
121 129
```

100100100100As with conventional vectors, conditions may be used to subset variables in a data frame. If for example, you want to check distribution of number of affected in men and women separately, you can do that by

```
> table(aff[sex == 1])

 0  1
98 31

> table(aff[sex == 0])

 0  1
96 25
```

On R command line pressing the "up-arrow" button makes the last typed command re-appear (pressing it one more time will bring you to the one before the last, so on). This is very handy when you have to repeat the same analysis of different variables

Summary:

- The list of available objects can be viewed with `ls()`; a class of some object `obj` can be interrogated with `class(obj)`.
- Simple summary statistics for numeric variables can be generated by using `summary` function
- Histogram for some variable `var` can be generated by `hist(var)`
- A variable with name `name` from a data frame `frame`, can be accessed through `frame$name`.
- You can attach the data frame to the search path by `attach(frame)`. Then the variables contained in this data frame may be accessed directly. To detach the data frame (because, e.g., you are now interested in other data frame), use `detach(frame)`.

Exercise 6 *An* `srdta` *data supplied with* `GenABEL` *contains a data frame with phenotypes. This data frame may be accessed through* `srdta@phdata`*. Explore this data frame and answer the questions*

1. *What is the value of the 4th variable for subject number 75?*
2. *What is the value of variable 1 for person 75? Check what is the value of this variable for the first ten people. Can you guess what variable 1 is?*
3. *What is sum of the variable 2? Can you guess what it is?*

Exercise 7 *Explore* `phdata` *slot of* `srdta`

1. *How many people has age over 65 years?*
2. *What is the sex distribution in the people over 65 years old?*

2.3 Exploratory analysis of qualitative and quantitative traits

Contingency tables of more then one variable may be generated in R using the `table` command we have being using before to explore frequency distributions. For example, if you want cross-tabulate sex and affection status in the data frame `assoc`, you can use

```
> table(sex, aff)

   aff
sex  0  1
  0 96 25
  1 98 31
```

Here, the first variable (sex) is presented in rows and the second (affection status) in columns.

As is usual case with R, the output may be saved as a new object (of class 'table', which is a variety of a matrix):

```
> a <- table(sex, aff)
> a

   aff
sex  0  1
  0 96 25
  1 98 31
```

and this object may be analysed further.

For example, we can easily get the number of affected male with

```
> a[2, 2]

[1] 31
```

Alternatively, we can analyse the resulting contingency table `a` with more complex functions. If we want to see proportions in this table, we can use

```
> prop.table(a)

   aff
sex     0     1
  0 0.384 0.100
  1 0.392 0.124
```

Needless to say, this is equivalent to

```
> prop.table(table(assoc$sex, assoc$aff))

        0     1
  0 0.384 0.100
  1 0.392 0.124
```

In the above table, we see what proportion of people belong to four different classes (affected male, affected female, unaffected male and unaffected female). We may be though interested in the proportion of males in affected and unaffected. This may be achieved by

```
> prop.table(a, 2)

   aff
sex         0         1
  0 0.4948454 0.4464286
  1 0.5051546 0.5535714
```

saying us that 55.4% of affected are male.

Alternatively, we may be interested in proportion of affected among males/females. To answer this question, run

```
> prop.table(a, 1)

   aff
sex         0         1
  0 0.7933884 0.2066116
  1 0.7596899 0.2403101
```

saying us that 55.4% of male are affected.

Other useful contingency table analysis function is `fisher.test`, which implements the Fisher Exact Test of independence:

```
> fisher.test(a)

        Fisher's Exact Test for Count Data

data:  a
p-value = 0.547
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.6409648 2.3156591
sample estimates:
odds ratio
  1.213747
```

Exploration of genetic data within base R, though possible, may be a bit of a pain. For example, we can easily generate contingency table of SNP5 vs affected status:

```
> a <- table(aff, snp5)
> a

   snp5
aff A/A B/A B/B
  0  31  88  71
  1   9  26  17
```

We can also look up what is the proportion of affected among different genotypic groups

```
> prop.table(a, 2)

   snp5
aff       A/A       B/A       B/B
  0 0.7750000 0.7719298 0.8068182
  1 0.2250000 0.2280702 0.1931818
```

showing that proportion of cases is similar in 'A/A' and 'A/B' genotypic groups and somewhat decreased in 'B/B'. It is easy to test if this affection is statistically independent of genotype by

```
> chisq.test(a)

        Pearson's Chi-squared test

data:  a
X-squared = 0.3874, df = 2, p-value = 0.8239
```

which gives (insignificant) genotypic association test on two degrees of freedom.

However, testing Hardy-Weinberg equilibrium, testing allelic effects, and even computation of allelic frequency is not so straightforward. Such specific genetic tests are implemented in special R libraries, such as `genetics` and `GenABEL` and will be covered in later sections of this document.

At this moment we will switch to exploratory analysis of quantitative traits. We will make use of the `srdta` data supplied with `GenABEL` . As you can remember from an exercise, the library is loaded with `library(GenABEL)` and the data are loaded with `data(srdta)`: Then the phenotypic data frame may be accessed through `srdta@phdata`.

Exercise 8 *Explore* `srdta@phdata`*. How many observations and variables are presented in the data frame? What are the classes of these variables?*

As it was mentioned before, the function `summary()` generates a summary statistics for an object. For example, to see summary for trait `qt1`, we can use

```
> summary(srdta@phdata$qt1)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
-4.6000 -0.9500 -0.3100 -0.2981  0.3800  3.2000  3.0000
```

`summary` **is quite useful function which may operate in different ways for objects of different classes. Try** `summary(srdta@phdata)`**.**

With R, it is also easy to explore the data graphically. For example, the histogram for `qt1` may be generated by

```
> hist(srdta@phdata$qt1)
```

(resulting histogram is shown at figure 1)

In similar manner, scatter-plots may be generated. To see relation between `qt1` and `qt3`, you can run

```
> plot(srdta@phdata$qt1, srdta@phdata$qt3)
```

(resulting plot is shown at figure 2)

The mean, median, minimum and maximum of the distribution of the trait may be found out using functions `mean`, `median`, `min` and `max`, respectively. The variance and standard deviation can be computed with `var` and `sd`.

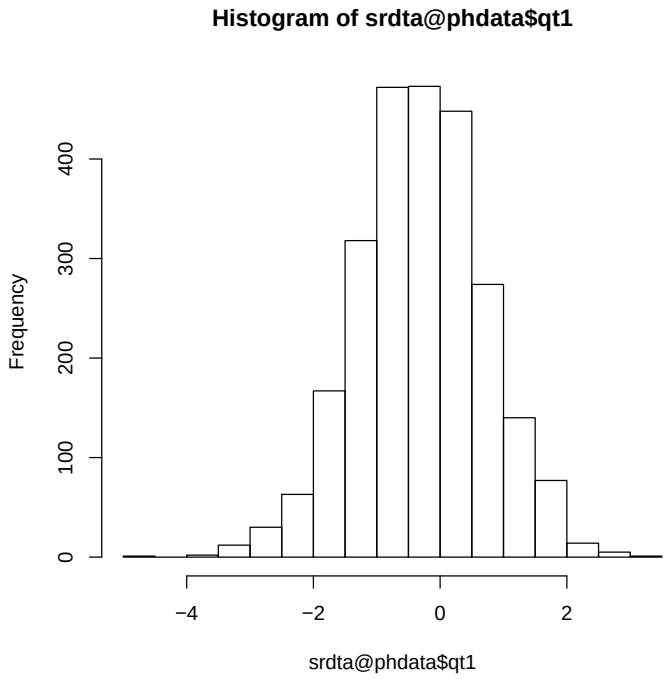

Figure 1: Histogram of qt1

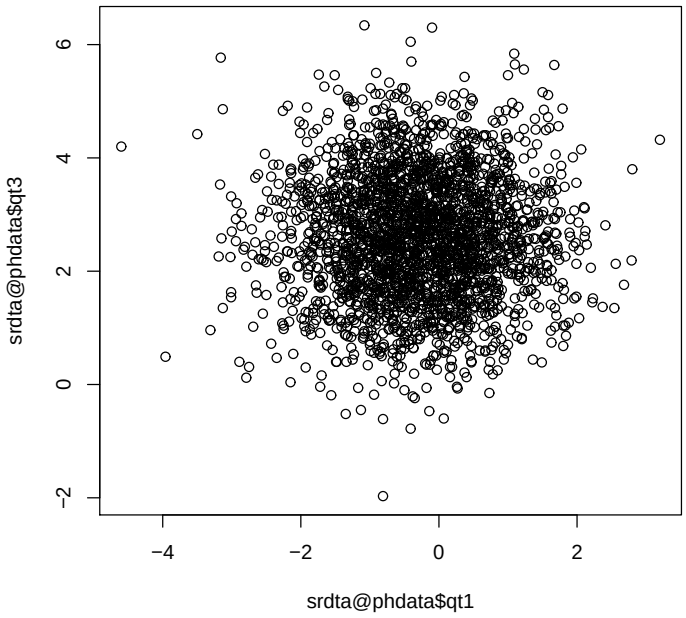

Figure 2: Scatter-plot of qt1 against qt3

To compute correlation between two variables (or all variables in a matrix/data frame), use `cor`.

In `GenABEL` , there is a special function designed to facilitate phenotypic quality control. This function takes names of variables and a data frame as an input, and returns summary statistics, list of outliers (using False Discovery Rate) and graphs.

For example, to do QC of sex, age and qt3, try

```
> check.trait(c("sex", "age", "qt3"), srdta@phdata)

--------------------------------
Trait sex has 2500 measurements
Missing: 0 ( 0 %)
Mean = 0.51 ; s.d. = 0.5
NO outliers discovered for trait sex
--------------------------------
Trait age has 2500 measurements
Missing: 0 ( 0 %)
Mean = 50.0378 ; s.d. = 7.060125
NO outliers discovered for trait age
--------------------------------
Trait qt3 has 2489 measurements
Missing: 11 ( 0.44 %)
Mean = 2.60859 ; s.d. = 1.101154
NO outliers discovered for trait qt3
```

The corresponding graph is depicted at figure **??**.

Before you start with the exercise: if a function returns unexpected results, and you are confident that syntax was right, checking help page is always a good idea!

Exercise 9 *Explore variables in* `phdata` *slot of* `srdta`

1. *What is the mean, median, minimum and maximum age in the sample?*
2. *Compare the distribution of qt3 in people younger and older than 65 years. Use function* `sd(A)` *to get standard deviation of A.*
3. *Produce distributions of different traits. Do you see something special?*
4. *What is correlation between qt3 and age?*

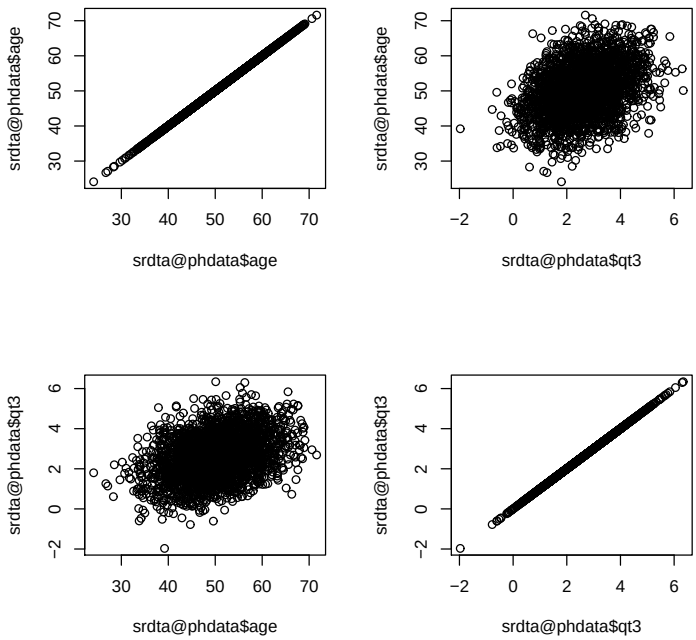

Figure 3: Quality control graph for `sex, age, qt3`

2.4 Regression analysis

While contingency tables, bi-plots and correlation are powerful tools to analyse relations between pairs of variable, a more general framework allowing investigation of relation of an outcome to multiple predictors is regression. In R, function `lm` implements linear regression modelling, and function `glm` implements generalised linear regression. In this section, we will use these two functions to analyse quantitative an binary outcomes.

You can do linear regression to check if trait `qt2` has relation with sex and age by

```
> a <- lm(srdta@phdata$qt2 ~ srdta@phdata$age + srdta@phdata$sex)
```

The results of analysis are stored in object 'a', which has class 'lm' and contains may sub-objects:

```
> class(a)

[1] "lm"

> names(a)

 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

At this moment you do not need to understand all these sub-objects; the meaningful summary of analysis is produced with

```
> summary(a)

Call:
lm(formula = srdta@phdata$qt2 ~ srdta@phdata$age + srdta@phdata$sex)

Residuals:
     Min       1Q   Median       3Q      Max
 -5.6498  -1.7953  -1.0328  -0.3148 883.0808

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      -1.55892    4.41667  -0.353    0.724
srdta@phdata$age  0.14022    0.08668   1.618    0.106
srdta@phdata$sex  1.30377    1.22393   1.065    0.287

Residual standard error: 30.59 on 2497 degrees of freedom
Multiple R-squared: 0.001518,        Adjusted R-squared: 0.0007181
F-statistic: 1.898 on 2 and 2497 DF,  p-value: 0.1501
```

You can see that `qt2` is not associated with age or sex.

As before, to make easy access to your data (basically, to avoid typing `srdta@phdata` before every trait name, you may attach the data to the search path:

```
> attach(srdta@phdata)
```

Then,the above expression to run linear regression analysis simplifies to:

```
> summary(lm(qt2 ~ age + sex))

Call:
lm(formula = qt2 ~ age + sex)

Residuals:
     Min       1Q   Median       3Q      Max
 -5.6498  -1.7953  -1.0328  -0.3148 883.0808

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.55892    4.41667  -0.353    0.724
age          0.14022    0.08668   1.618    0.106
sex          1.30377    1.22393   1.065    0.287

Residual standard error: 30.59 on 2497 degrees of freedom
Multiple R-squared: 0.001518,        Adjusted R-squared: 0.0007181
F-statistic: 1.898 on 2 and 2497 DF,  p-value: 0.1501
```

with the same results.

Analysis of binary outcomes may be performed using `glm` function, using *binomial family* for the error distribution and the link function. For example, to figure out if your binary trait (`bt`) is associated with sex and age, you need to tell that this is binary trait:

```
> a <- glm(bt ~ age + sex, family = "binomial")
> summary(a)

Call:
glm(formula = bt ~ age + sex, family = "binomial")

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.992  -1.091  -0.444   1.094   1.917

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.639958   0.330519 -14.038  < 2e-16 ***
age          0.088860   0.006463  13.749  < 2e-16 ***
sex          0.379593   0.084138   4.512 6.44e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3450.5  on 2488  degrees of freedom
Residual deviance: 3216.5  on 2486  degrees of freedom
  (11 observations deleted due to missingness)
AIC: 3222.5

Number of Fisher Scoring iterations: 4
```

There is strong association between `bt` and sex and age. If you want to characterise the strength of association to a binary trait with Odds Ratios, take the exponents of the regression coefficient. For example, the odds ratio associated with male is

```
> exp(0.3796)

[1] 1.461700
```

3 Introduction to association analysis in R

When analyzing several (dozens of) SNPs, facilities of base R are sufficient and efficient for data storage and analysis. Few specific test, such as these of Hardy-Weinberg Equilibrium (HWE) and Linkage Disequilibrium (LD), are implemented in different libraries, e.g. `genetics` and `GenABEL` .

In this section, we will describe library `genetics` and will make use of it to guide you through simple genetic analysis exercise using a small example data set. In the last part, you will investigate a bigger data set as based on the knowledge obtained in the first part, and will answer the questions.

3.1 Exploring genetic data with library genetics

Library `genetics` was written by Gregory R. Warnes to facilitate analysis of genetic data in R. This library

- Implements genetic analysis tests, such as test for Hardy-Weinberg equilibrium and Linkage disequilibrium.
- Implements new data classes, such as `genotype`, `haplotype` and `LD.data.frame`.
- Modifies default R functions, such as `summary` and `plot` to correctly analyse and present these new classes.
- Facilitates export of the data from R to the formats supported by such genetic analysis packages as GenePop and QTDT.

Start `R` by double-click on the file `ge03d1p1.RData`. Load library `genetics`, which we will need for testing HWE and computations of LD by

```
> library(genetics)
```

The file you have loaded contains single data frame `assocg`. Let us briefly explore it:

```
> class(assocg)

[1] "data.frame"

> names(assocg)

[1] "subj" "sex"  "aff"  "qt"   "snp4" "snp5" "snp6"

> dim(assocg)

[1] 250   7
```

You can see that `assocg` looks remarkably similar to the previously explored data frame `assoc` (section 2.2, page 11). Indeed, they are almost equivalent. Let us present the data for the subjects 5 to 15 and compare this output to that presented on page 13:

```
> assocg[5:15, ]

     subj sex aff         qt snp4 snp5 snp6
1533 1533   0   0  0.1009220  A/B  B/A  B/A
2466 2466   1   0 -0.1724321  A/B  A/A  A/A
2425 2425   0   0 -0.3378473  B/B  A/A  A/A
1068 1068   0   0 -1.7112925  A/A  B/B <NA>
198   198   1   0 -0.4815822  A/B  B/A  B/A
1496 1496   1   0  1.2281232  A/A  B/B  B/B
909   909   0   0  0.5993945  A/B  B/A  B/A
1213 1213   0   0  1.9792190  A/A  B/B  B/B
181   181   1   0  1.5435921  A/A  B/B  B/B
1783 1783   0   0 -1.6242738  A/B  B/A  B/A
1914 1914   0   0 -0.5160331  A/A  B/B  B/B
```

The data are identical. However, the SNP data presented in the new data frame have special class `genotype`, as implemented in `genetics` library:

```
> class(assocg$snp4)

[1] "genotype" "factor"
```

Previously, the SNP genotypes were coded as characters. This new way of presentation allows library `genetics` to recognise the SNP data as genetic and analyse them accordingly.

Let us attach the `assocg` data frame and explore what data analysis advantages are achieved by application of library `genetics`.

```
> attach(assocg)
```

As we noted in section 2.2, testing Hardy-Weinberg equilibrium, testing allelic effects, and even computation of allelic frequency is not so straightforward in base R. These tests, are, however, easy with library `genetics`. To see the allelic frequencies and other summary statistics for a SNP, you can use

```
> summary(snp4)

Number of samples typed: 243 (97.2%)

Allele Frequency: (2 alleles)
   Count Proportion
A    323       0.66
B    163       0.34
NA    14         NA


Genotype Frequency:
    Count Proportion
B/B    29       0.12
A/B   105       0.43
A/A   109       0.45
NA      7         NA

Heterozygosity (Hu)  = 0.4467269
Poly. Inf. Content   = 0.3464355
```

To check these characteristics in controls and cases separately, you can use

```
> summary(snp4[aff == 0])

Number of samples typed: 190 (97.9%)

Allele Frequency: (2 alleles)
   Count Proportion
A    255       0.67
B    125       0.33
NA     8         NA
```

```
Genotype Frequency:
    Count Proportion
B/B    22       0.12
A/B    81       0.43
A/A    87       0.46
NA      4         NA

Heterozygosity (Hu)  = 0.4426469
Poly. Inf. Content   = 0.3440288
```

```
> summary(snp4[aff == 1])

Number of samples typed: 53 (94.6%)

Allele Frequency: (2 alleles)
   Count Proportion
A     68       0.64
B     38       0.36
NA     6         NA


Genotype Frequency:
    Count Proportion
B/B     7       0.13
A/B    24       0.45
A/A    22       0.42
NA      3         NA

Heterozygosity (Hu)  = 0.4643306
Poly. Inf. Content   = 0.3541731
```

Let us check if HWE holds for the SNPs described in this data frame. We can do exact test for HWE by

```
> HWE.exact(snp4)

        Exact Test for Hardy-Weinberg Equilibrium

data:  snp4
N11 = 109, N12 = 105, N22 = 29, N1 = 323, N2 = 163, p-value = 0.666
```

If you want to check HWE using controls only, you can do it by

```
> HWE.exact(snp4[aff == 0])

        Exact Test for Hardy-Weinberg Equilibrium

data:  snp4[aff == 0]
N11 = 87, N12 = 81, N22 = 22, N1 = 255, N2 = 125, p-value = 0.6244
```

Let us check if the there is LD between snp4 and snp5:

```
> LD(snp4, snp5)

Pairwise LD
-----------
                    D        D'      Corr
Estimates: 0.2009042 0.9997352 0.8683117

              X^2 P-value   N
LD Test: 354.3636       0 235
```

The output shows results of the test for significance of LD, and estimates of the magnitude of LD (D' and correlation, r). To obtain r^2, you can either square the correlation manually

```
> 0.8683117 * 0.8683117

[1] 0.7539652
```

or simply ask `LD()` to report it by

```
> LD(snp4, snp5)$"R^2"

[1] 0.7539652
```

The latter command is possible because the `LD()` function actually computes more things than it reports. This is quite common for `R` functions. You can apply `names()` function to the analysis objects to see (at least part of) what was actually computed. Try

```
> ld45 <- LD(snp4, snp5)
```

and check what are the sub-objects contained in this analysis object

```
> names(ld45)

[1] "call"    "D"       "D'"      "r"       "R^2"     "n"       "X^2"
[8] "P-value"
```

Any of these variables can be accessed through `object$var` syntax, e.g. to check D' we can use

```
> ld45$"D'"

[1] 0.9997352
```

To check LD for more that two SNPs, we can compute an LD analysis object by

```
> ldall <- LD(data.frame(snp4, snp5, snp6))
```

and later check

```
> ldall$"P-value"

     snp4 snp5 snp6
snp4   NA    0    0
snp5   NA   NA    0
snp6   NA   NA   NA
```

to see significance,

```
> ldall$"D'"
```

```
     snp4      snp5      snp6
snp4   NA 0.9997352 0.8039577
snp5   NA        NA 0.9997231
snp6   NA        NA        NA
```

for D' and

```
> ldall$"R^2"
```

```
     snp4      snp5      snp6
snp4   NA 0.7539652 0.5886602
snp5   NA        NA 0.8278328
snp6   NA        NA        NA
```

for r^2.

You can also present e.g. r^2 matrix as a plot by

```
> image(ldall$"R^2")
```

A more neat way to present it requires specification of the set of threshold (break points) and colors to be used (you do not need to try this example if you do not want):

```
> image(ldall$"R^2", breaks = c(0.5, 0.6, 0.7, 0.8, 0.9, 1), col = heat.colors(5))
```

Resulting plot is shown at figure 4.

For any `R` command, you can get help by typing `help(command)`. Try `help(image)` if you are interested to understand what are "breaks" and "col"; or try `help(heat.colors)` to figure this color schema out.

Similar to our HWE checks, we may want to compute (and compare) LD in cases and controls separately:

```
> ldcases <- LD(data.frame(snp4, snp5, snp6)[aff == 1, ])
> ldcases$"R^2"
```

```
     snp4      snp5      snp6
snp4   NA 0.7615923 0.6891558
snp5   NA        NA 0.8943495
snp6   NA        NA        NA
```

```
> ldcontr <- LD(data.frame(snp4, snp5, snp6)[aff == 0, ])
> ldcontr$"R^2"
```

```
     snp4      snp5      snp6
snp4   NA 0.7512458 0.5616395
snp5   NA        NA 0.8075894
snp6   NA        NA        NA
```

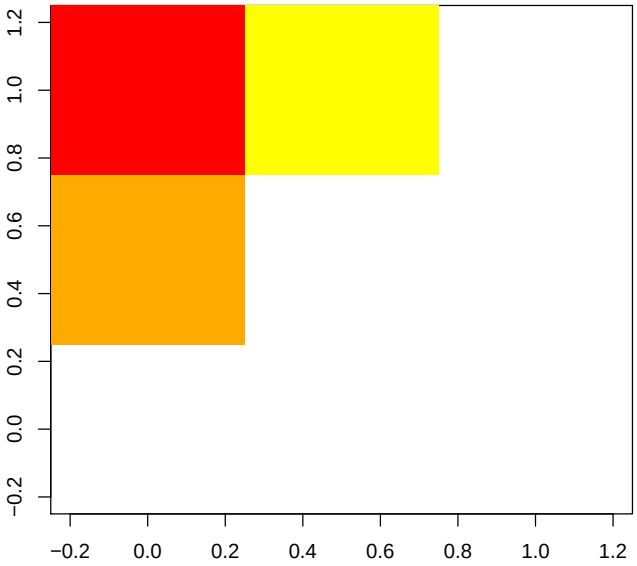

Figure 4: r^2 plot for snp4, snp5 and snp6

and even present it results for cases and controls on the same graph (you do not need to produce this graph, which is presented at the figure 5):

```
> image(ldcases$"R^2", breaks = c(0.5, 0.6, 0.7, 0.8, 0.9, 1),
+     col = heat.colors(5))
> image(t(ldcontr$"R^2"), breaks = c(0.5, 0.6, 0.7, 0.8, 0.9, 1),
+     col = heat.colors(5), add = T)
```

3.2 Example association analysis

Now, after we have described genetic and phenotypic data separately, we are ready to test association between these two. In previous sections, we showed that association between a binary trait and genotype may be analysed using contingency tables (functions `table`, `prop.table`, `fisher.test`, etc.). The association between a quantitative trait and genotype may be done using correlations, T-test, etc.

However, a more flexible analysis is possible when using regression modelling. First, we will investigate relation between the quantitative trait `qt` and the SNPs by using linear regression

```
> mg <- lm(qt ~ snp4)
```

The `lm` command fits linear regression model to the data and returns an analysis object. The summary of analysis may be generated with

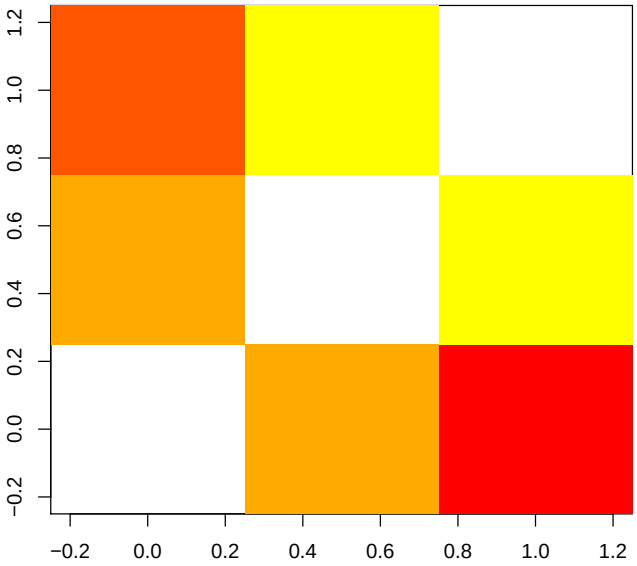

Figure 5: r^2 plot for snp4, snp5 and snp6. Above diagonal: LD in cases; below: controls

```
> summary(mg)

Call:
lm(formula = qt ~ snp4)

Residuals:
     Min       1Q   Median       3Q      Max
-2.63700 -0.62291 -0.01225  0.58922  3.05561

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.081114   0.092517  -0.877    0.382
snp4A/B     -0.108366   0.132079  -0.820    0.413
snp4B/B     -0.006041   0.201820  -0.030    0.976

Residual standard error: 0.9659 on 240 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared: 0.003049,        Adjusted R-squared: -0.005259
F-statistic: 0.367 on 2 and 240 DF,  p-value: 0.6932
```

From the summary output, it is clear that the model assumes arbitrary (estimated) effects of the genotypes AA, AB and BB. Neither effect of AB nor BB is significant in this case. The global test on two degrees of freedom (bottom of

the output) is also not significant.

If you want to include some covariate into your model, e.g. sex, you can easily do that by adding the term to the formula:

```
> summary(lm(qt ~ sex + snp4))

Call:
lm(formula = qt ~ sex + snp4)

Residuals:
      Min        1Q    Median        3Q       Max
-2.664422 -0.624169 -0.008752  0.597045  3.080857

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.110298   0.115260  -0.957    0.340
sex          0.053018   0.124493   0.426    0.671
snp4A/B     -0.104429   0.132628  -0.787    0.432
snp4B/B     -0.002452   0.202340  -0.012    0.990

Residual standard error: 0.9676 on 239 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared: 0.003805,        Adjusted R-squared: -0.0087
F-statistic: 0.3043 on 3 and 239 DF,  p-value: 0.8223
```

You can also allow for interaction by using the "*" operator

```
> summary(lm(qt ~ sex * snp4))

Call:
lm(formula = qt ~ sex * snp4)

Residuals:
      Min        1Q    Median        3Q       Max
-2.570485 -0.645961 -0.002641  0.610938  3.019696

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.20579    0.13834  -1.487    0.138
sex          0.22649    0.18647   1.215    0.226
snp4A/B      0.05222    0.19024   0.274    0.784
snp4B/B      0.18071    0.28576   0.632    0.528
sex:snp4A/B -0.30191    0.26566  -1.136    0.257
sex:snp4B/B -0.35508    0.40531  -0.876    0.382

Residual standard error: 0.9684 on 237 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared: 0.01041,        Adjusted R-squared: -0.01047
F-statistic: 0.4984 on 5 and 237 DF,  p-value: 0.7773
```

Note that both main effects of sex and snp4, and also effects of interaction are estimated in this model.

Of interest in genetic studies may be three other models: additive, dominant and recessive.

The additive model assumes that the difference between mean trait's values between 'AA' and 'BB' is twice the difference between 'AA' and 'BB', that is the mean value of the trait for heterozygous genotypes is right in between the two homozygotes. To test additive model, we first need to recode the predictor (genotype) as a numeric factor to be used as covariate. This can be easy done with function `as.numeric`:

```
> add4 <- as.numeric(snp4) - 1
```

We can check if recoding was done correctly by producing the table

```
> table(snp4, add4)
```

```
     add4
snp4    0   1   2
  A/A 109   0   0
  A/B   0 105   0
  B/B   0   0  29
```

Now to test the additive model run

```
> summary(lm(qt ~ add4))
```

```
Call:
lm(formula = qt ~ add4)

Residuals:
     Min       1Q   Median       3Q      Max
-2.54813 -0.62104 -0.02754  0.60584  3.00652

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.10476    0.08710  -1.203    0.230
add4        -0.03563    0.09133  -0.390    0.697

Residual standard error: 0.9651 on 241 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared: 0.0006313,        Adjusted R-squared: -0.003516
F-statistic: 0.1522 on 1 and 241 DF,  p-value: 0.6968
```

The model assuming dominant action of the 'A' allele means that the means of genotypes 'AA' and 'AB' are the same. This is equivalent to the model of recessive action of 'B' allele. To code SNP4 according to this model, we can use function `replace`:

```
> dom4 <- add4
> dom4[dom4 == 2] <- 1
> table(snp4, dom4)
```

```
     dom4
snp4    0   1
```

```
  A/A 109   0
  A/B   0 105
  B/B   0  29
```

To test association with a binary outcome, we will use function `glm` with `binomial` family:

```
> summary(glm(aff ~ snp4, family = "binomial"))

Call:
glm(formula = aff ~ snp4, family = "binomial")

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.7433  -0.7204  -0.6715  -0.6715   1.7890

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.3749     0.2386  -5.761 8.35e-09 ***
snp4A/B       0.1585     0.3331   0.476    0.634
snp4B/B       0.2297     0.4952   0.464    0.643
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 254.91  on 242  degrees of freedom
Residual deviance: 254.58  on 240  degrees of freedom
  (7 observations deleted due to missingness)
AIC: 260.58

Number of Fisher Scoring iterations: 4
```

To make a test of global significance of the SNP effect, you can use

```
> anova(glm(aff ~ snp4, family = "binomial"), test = "Chisq")

Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    242    254.908
snp4   2    0.329       240    254.579     0.848
```

In the manner similar to that described for quantitative traits, additive and dominance/recessive models can be tested by proper coding of the genotypic variable, e.g. to test the additive model, use

```
> summary(glm(aff ~ as.numeric(snp4), family = "binomial"))

Call:
glm(formula = aff ~ as.numeric(snp4), family = "binomial")

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.7548  -0.7139  -0.6747  -0.6747   1.7842

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)       -1.4913     0.4164  -3.581 0.000342 ***
as.numeric(snp4)   0.1272     0.2268   0.561 0.574994
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 254.91  on 242  degrees of freedom
Residual deviance: 254.60  on 241  degrees of freedom
  (7 observations deleted due to missingness)
AIC: 258.60

Number of Fisher Scoring iterations: 4
```

Now you have learned all commands necessary to answer the questions of the next section.

Exit R by typing `q()` command (do not save image) and and proceed to the self exercise.

3.3 Exercise

Start `R` by double-click over the file `ge03d1p2.RData`. Explore the data frame present and answer the questions.

Exercise 10 *How many SNPs are described in this data frame?*

Exercise 11 *What is the frequency (proportion) of snp1 allele A? What is its frequency in these affected (aff==1)?*

Exercise 12 *How many cases and controls are present?*

Exercise 13 *If all subjects are used to test HWE, are there any SNPs out of HWE at nominal $P \leq 0.05$? Which ones?*

Exercise 14 *If only controls are used to test the SNPs which are out of HWE in total sample, are these still out of HWE?*

Exercise 15 *Which SNP pairs are in strong LD ($r^2 \geq 0.8$)?*

Exercise 16 *For SNPs in strong LD, what is r^2 for separate samples of cases and controls?*

Exercise 17 *Is there significant association between affection status and sex? What is P-value for association?*

Exercise 18 *Is association between the disease and* `qt` *significant?*

Exercise 19 *Which SNPs are associated with the quantitative trait* `qt` *at nominal* $P \leq 0.05$*? Use 2 d.f. test.*

Exercise 20 *Test each SNP for association with the affection status, using 2 d.f. test. Which SNPs are significantly associated at nominal* $P \leq 0.05$*? How can you describe the model of action of the significant SNPs?*

Exercise 21 *For the SNPs selected in previous question, test association using additive model. Which SNPs are still associated?*

Exercise 22 *If you adjust the analysis under additive model (question 21) for significant covariates which you discovered in questions 17 and 18, are these findings still significant?*

Exercise 23 *Test association between* `aff` *and snp5 and snp10, allowing for the SNPs interaction effect. Use arbitrary (not an additive) model. Do you observe significant interaction? How can you describe the model of concert action of snp5 and snp10?*

4 Introduction to GenABEL

In this section, you will become familiar with the `GenABEL` library, designed for GWA analysis. Compared to `genetics` package, it provides specific facilities for storage and manipulation of large amounts of data, very fast tests for GWA analysis, and special functions to analyse and graphically present the results of GWA analysis (thus "analysis of analysis").

Start R and load `GenABEL` library using command

```
> library(GenABEL)
```

After that, load the data with the command

```
> data(srdta)
```

4.1 General description of `gwaa.data-class`

The object you have loaded, `srdta`, belongs to the `gwaa.data` class. This is a special class developed to facilitate GWA analysis.

In GWA analysis, different types of data are used. These include the phenotypic and genotypic data on the study participants and chromosome and location of every SNP. For every SNP, it is desirable to know the details of coding (what are alleles? – A, T, G, C? – and what is the strand – '+' or '-', 'top' or 'bot'? – this coding is for).

One could attempt to store all phenotypes and genotypes together in a single table, using, e.g. one row per study subject; than the columns will correspond to study phenotypes and SNPs. For a typical GWA data set, this would lead

to a table of few thousands rows and few hundreds of thousands of columns. Such a format is generated when one downloads HapMap data for a region. To store GWA data in such tables internally, within R, proves to be inefficient. In `GenABEL`, special data class, `gwaa.data-class` is used to store GWA data. The structure of this data class is shown at the figure 6.

An object of some class has "slots" which may contain actual data or objects of other classes. The information stored at a particular `slot` of an `object` can be accessed by command `object@slot`.

At the first level, a `gwaa.data-class` object has slot `phdata`, which contains all phenotypic information in a data frame (`data.frame-class` object). The rows of this data frame correspond to study subjects, and the columns correspond to the variables. There are two default variables, which are always present in `phdata`. The first of these is "id", which contains study subject identification code. This identification code can be arbitrary character, but every person must be coded with an unique ID. The second default variable is "sex", where males are coded with ones ("1") and females are coded with zero ("0"). It is important to understand that this data frame is not supposed to be directly modified by the user. In particular, it is extremely important to remember that one should not directly add subjects to the table, change the values of "id" and "sex", and change the order of subjects in `phdata` unless this one is really understands the way `GenABEL` works. One also should not run such data manipulation functions as `merge`, `cbind` and `rbind` – exactly because they may change the number of subjects or interfere with the order. On the other hand, it is OK to add more variables to the data frame through direct computations, for example, if one wishes to add computed body mass index, it is OK to run the command like

```
obj@phdata$bmi <- obj@phdata$weight/((obj@phdata$height)^2)
```

To add many variables to `phdata`, special GenABEL function `add.phdata` should be used.

The other slot of an object of `gwaa.data-class` is slot `gtdata`, which contains all GWA genetic information in an object of class `snp.data` class (figure 6). This class, in turn, has slots `nids`, containing the number of study subjects, `idnames`, containing all ID names of these subjects, `nsnps`, containing the number of SNPs typed, `snpnames`, containing the SNP names, `chromosome`, containing the name of the chromosome the SNPs belong to and slot `map` with map position of SNPs, and slot `male`, containing the sex code for the subjects (1=male, 0=female). The latter is identical to the "sex" variable contained in the `phdata`, but is duplicated here because many operations with purely genetic data, in particular these concerning analysis of sex chromosomes, depend on the sex. The strand information is presented in the slot `strand`. `GenABEL` codes strand as "+" (forward), "-" (reverse) or "u" (unknown). Of cause, if you prefer top/bottom coding, this information may be stored in the same form – you will just need to remember that "+" corresponds to e.g. "top", and "-" to "bottom" strand. The allelic coding is presented in slot `coding`. Coding for every allele is presented with a pair of characters, for example "AG". Thus, for such polymorphism, you may expect "AA", "AG" and "GG" genotypes to be found in population. The order (that is "AG" vs "GA") is important – the first allele reported is the one which will be used as a reference in association analysis, and thus the effects are reported for the second allele. To avoid memory overheads, the strand and coding information is internally stored as `snp.strand-class` and `snp.coding-class`. Information can be converted to human-readable format

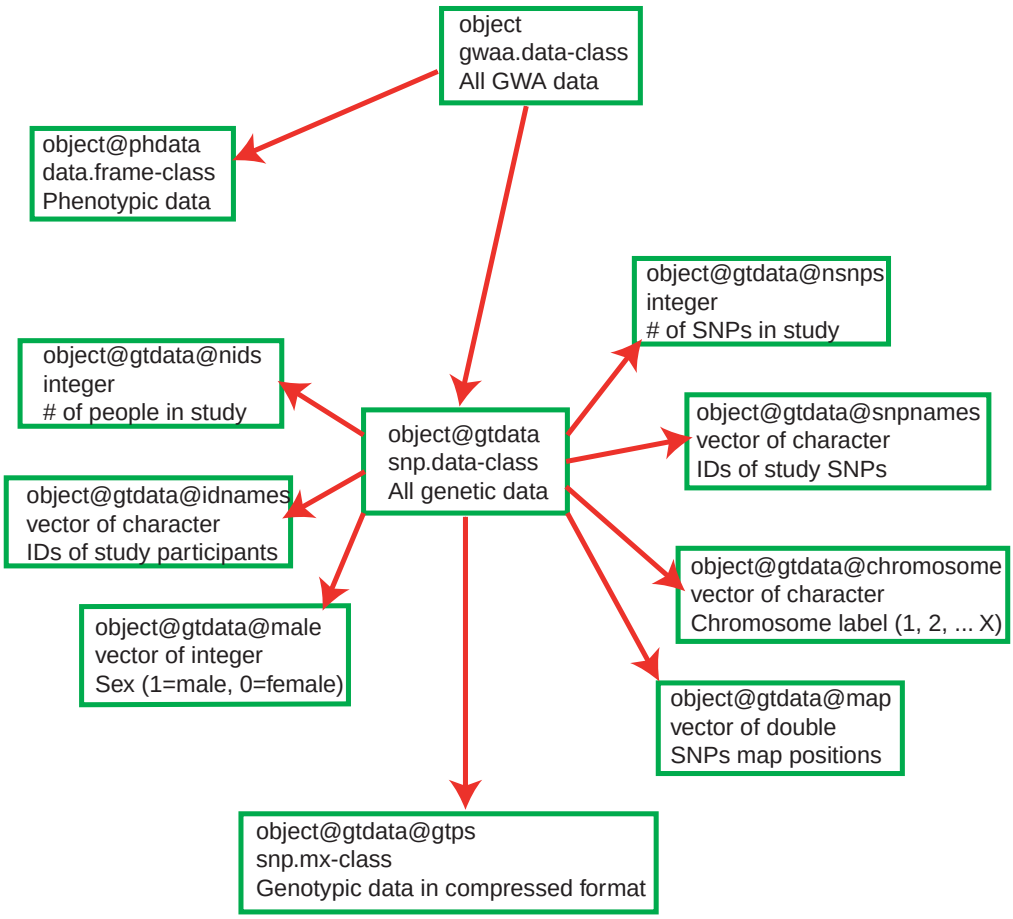

Figure 6: Structure of gwaa.data-class. In every box, first line contains the object and slot names, second line describes the class of this object, and third line describes what information is contained. NEEDS TO BE UPDATED WITH `object@gtdata@coding; snp.coding-class; snp allele coding` and `object@gtdata@strand; snp.strand-class; snp allele strand`

using `as.character` function.

If, for example, you would like to know, how many SNPs were included in the study (slot `nsnps` of the slot `gtdata` of `srdta`), you need to run command

```
> srdta@gtdata@nsnps

[1] 833
```

Thus, 833 SNPs were typed in the study. You can access information stored in any slot in this manner.

You may want to read the general `GenABEL` man page using `help(GenABEL)`. To see help on `gwaa.data-class`, you can use `help("gwaa.data-class")` (mind the quotation marks!).

Summary:

- An object of some class has "slots" which may contain actual data or objects of other classes. The information stored at a particular `slot` of an `object` can be accessed by command `object@slot`.
- `GenABEL` uses special data class, `gwaa.data-class`, to store GWA data.

Exercise 24 *Explore* `srdta`*.*

1. *How many people are included in the study?*
2. *How many of these are males?*
3. *How many are females?*
4. *What is male proportion?*

Exercise 25 *Explore slot containing map (*`map`*) and slot containing SNP names (*`snpnames`*) of the* `gtdata` *slot of* `srdta`*.*

1. *What are names of markers located after 2,490,000 b.p.?*
2. *Between 1,100,000 and 1,105,000 b.p.?*

4.2 Sub-setting and coercing gwaa.data

It is possible to sub-set the object, which stores the GWA data in the manner similar to the described above. Very primitively, you may think of an object of class `gwaa.data` as a matrix whose rows correspond to study subjects and columns correspond to SNPs studied (though the actual object is a way more complicated). For example, if we would like to investigate what is the content of `srdta` for the first 5 people and 3 SNPs, we can run

```
> ssubs <- srdta[1:5, 1:3]
> class(ssubs)

[1] "gwaa.data"
attr(,"package")
[1] "GenABEL"

> ssubs

  id sex  age   qt1    qt2  qt3 bt
1 p1   1 43.4 -0.58   4.46 1.43  0
2 p2   1 48.2  0.80   6.32 3.90  1
3 p3   0 37.9 -0.52   3.26 5.05  1
4 p4   1 53.8 -1.55 888.00 3.76  1
5 p5   1 47.5  0.25   5.70 2.89  1
@nids = 5
@nsnps = 3
@nbytes = 2
@idnames = p1 p2 p3 p4 p5
@snpnames = rs10 rs18 rs29
@chromosome = 1 1 1
@coding =  08 0b 0c
@strand =  01 01 02
@map = 2500 3500 5750
@male = 1 1 0 1 1
@gtps =
40 40 40
40 40 00
```

As you can see, by sub-setting we obtained a smaller object of `gwaa.data-class`, with all its slots. Most of the information is straightforward and does not need further explanation.

There are though three slots which are not human-readable, i.e. `@coding`, `@strand`, and `@gtps`. These are coded using new classes based in `R raw` data type; these can be converted in human-readable format using a variety of 'as.*' functions. For example, to see human-readable information on coding and strand, let us try

```
> as.character(ssubs@gtdata@coding)

rs10 rs18 rs29
"TG" "GA" "GT"
```

This tells what alleles are observed in the three SNPs, and which alleles will be used as a reference in association analysis.

To see the strand, use

```
> as.character(ssubs@gtdata@strand)

rs10 rs18 rs29
 "+"  "+"  "-"
```

The slot `gtps` contains the SNP data, and is not readable, because the information is compressed (each element contains data on up to four genotypes). To get human-readable information, an object of class `snp.data-class` (e.g. `srdta@gtdata`) can be coerced to a character using the same `as.character()` function:

```
> as.character(ssubs@gtdata)

   rs10  rs18  rs29
p1 "T/T" "G/G" "G/G"
p2 "T/T" "G/G" NA
p3 "T/T" "G/G" NA
p4 "T/T" "G/G" NA
p5 "T/T" "G/A" "G/G"
```

For `@gtps` conversion to other formats are possible as well. Other useful coercion is to "numeric":

```
> as.numeric(ssubs@gtdata)

   rs10 rs18 rs29
p1    0    0    0
p2    0    0   NA
p3    0    0   NA
p4    0    0   NA
p5    0    1    0
```

You can see that conversion to numeric data type uses information on both underlying genotypes and coding. When coding is "GA", as is for the `rs18` (the second SNP), homozygotes for the first allele, as specified by coding ("G") are converted to zeros ("0"), heterozygotes are converted to ones ("1"), and homozygotes for the second allele ("A") are converted to twos ("2"). Clearly, when numerically converted data are used for association analysis, the effects will be estimated for the second allele, while first will be used as a reference.

Genotypic data converted to standard R "numeric" format can be used in any further analysis. Homozygotes of one type are coded as "0", heterozygotes are coded as "1" and other type of homozygotes is coded as "2". You can think of this as the number of allele of "B" type.

Several useful genetic analysis libraries were developed for R. These include `genetics` (analysis of linkage disequilibrium and many other useful functions) and `haplo.stats` (analysis of association between traits and haplotypes). These use there own genetic data formats.

One can translate `GenABEL` genetic data to the format used by "genetics" library by `as.genotype()`:

```
> as.genotype(ssubs@gtdata)

   rs10 rs18 rs29
p1  T/T  G/G  G/G
p2  T/T  G/G <NA>
p3  T/T  G/G <NA>
p4  T/T  G/G <NA>
p5  T/T  G/A  G/G
```

To translate `GenABEL` data to the format used by "haplo.stats" you can use function `as.hsgeno()`

```
> as.hsgeno(ssubs@gtdata)

   rs10.a1 rs10.a2 rs18.a1 rs18.a2 rs29.a1 rs29.a2
p1       1       1       1       1       1       1
p2       1       1       1       1      NA      NA
p3       1       1       1       1      NA      NA
p4       1       1       1       1      NA      NA
p5       1       1       1       2       1       1
```

Actually, most users will not need the latter function, as `GenABEL` provides a functional interface to "haplo.stats" (such `GenABEL` functions as `scan.haplo()` and `scan.haplo.2D()`).

It is possible to select sub-sets of `gwaa.data-class` based not only on index (e.g. first 10 people and SNP number 33), but also based on names.

For example, if we would like to retrieve phenotypic data on people with IDs "p141", "p147" and "p2000", we can use

```
> srdta[c("p141", "p147", "p2000"), ]@phdata

        id sex  age   qt1  qt2  qt3 bt
141   p141   0 47.2  0.51 5.23 2.17  0
147   p147   0 43.2  0.14 4.47 1.73  0
2000 p2000   0 43.1 -1.53 2.78 2.70  1
```

here, the first part of expression sub-sets `srdta` on selected IDs, and the second tells which part of the retrieved sub-set we want to see. You can try `srdta[c("p141","p147","p2000"),]`, but be prepared to see long output, as all information will be reported.

In similar manner, we can also select on SNP name. For example, if we are interested to see information on SNPs "rs10" and "rs29" for above people, we can run

```
> srdta[c("p141", "p147", "p2000"), c("rs10", "rs29")]

        id sex  age   qt1  qt2  qt3 bt
141   p141   0 47.2  0.51 5.23 2.17  0
147   p147   0 43.2  0.14 4.47 1.73  0
2000 p2000   0 43.1 -1.53 2.78 2.70  1
@nids = 3
@nsnps = 2
```

```
@nbytes = 1
@idnames = p141 p147 p2000
@snpnames = rs10 rs29
@chromosome = 1 1
@coding =  08 0c
@strand =  01 02
@map = 2500 5750
@male = 0 0 0
@gtps =
40 40
```

To see the actual genotypes for the above three people and two SNPs, use

```
> as.character(srdta[c("p141", "p147", "p2000"), c("rs10", "rs29")])
```

```
      rs10  rs29
p141  "T/T" "G/G"
p147  "T/T" "G/G"
p2000 "T/G" "G/T"
```

or

```
> as.numeric(srdta[c("p141", "p147", "p2000"), c("rs10", "rs29")])
```

```
      rs10 rs29
p141     0    0
p147     0    0
p2000    1    1
```

Exercise 26 *Explore genotypes for SNP "rs114".*

1. *What is the coding and which allele is the reference one?*
2. *What is the frequency of* ***non-reference*** *("effective") allele in total sample?*
3. *What is the frequency of effective allele in male?*
4. *What is the frequency of effective allele in female?*
5. *What is the frequency of the* ***reference*** *allele in total sample, males and females?*

Summary:

- It is possible to obtain subsets of objects of `gwaa.data-class` and `snp.data-class` using standard 2D sub-setting model `[i,j]`, where `i` corresponds to study subjects and `j` corresponds to SNPs.

- It is possible to provide ID and SNP names instead of indexes when subsetting an object of class `gwaa.data-class`.
- Function `as.numeric()` converts genotypic data from `snp.data-class` to regular integer numbers, which can be used in analysis with R.
- Function `as.character()` converts genotypic data from `snp.data-class` to character format.
- Function `as.genotype()` converts genotypic data from `snp.data-class` to the format used by library `genetics`.
- Function `as.hsgeno()` converts genotypic data from `snp.data-class` to the format used by library `haplo.stats`.

4.3 Exploring genetic data

Implementation of function `summary()` to `snp.data` class is very useful in genetic data exploration and quality control (QC). Let us try application of this function to the `ssubs`:

```
> a <- summary(ssubs)
> a

$phdata
      id                 sex           age             qt1
 Length:5           Min.   :0.0   Min.   :37.90   Min.   :-1.55
 Class :character   1st Qu.:1.0   1st Qu.:43.40   1st Qu.:-0.58
 Mode  :character   Median :1.0   Median :47.50   Median :-0.52
                    Mean   :0.8   Mean   :46.16   Mean   :-0.32
                    3rd Qu.:1.0   3rd Qu.:48.20   3rd Qu.: 0.25
                    Max.   :1.0   Max.   :53.80   Max.   : 0.80
      qt2              qt3             bt
 Min.   :  3.26   Min.   :1.430   Min.   :0.0
 1st Qu.:  4.46   1st Qu.:2.890   1st Qu.:1.0
 Median :  5.70   Median :3.760   Median :1.0
 Mean   :181.55   Mean   :3.406   Mean   :0.8
 3rd Qu.:  6.32   3rd Qu.:3.900   3rd Qu.:1.0
 Max.   :888.00   Max.   :5.050   Max.   :1.0

$gtdata
     NoMeasured CallRate Q.2 P.11 P.12 P.22 Pexact       Fmax      Plrt
rs10          5      1.0 0.0    5    0    0      1  0.0000000 1.0000000
rs18          5      1.0 0.1    4    1    0      1 -0.1111111 0.7386227
rs29          2      0.4 0.0    2    0    0      1  0.0000000 1.0000000
     Chromosome
rs10          1
rs18          1
rs29          1
```

In the first section, the summary is generated for phenotypic data. In the second section, summary is generated for genotypic data. In this section, `NoMeasured` refers to the number of genotypes scores, `CallRate` to the proportion of these, `Q.2` is the frequency of the 'B' allele. The counts in three genotypic classes are provided next. `Pexact` refers to exact P-value for the test of Hardy-Weinberg equilibrium.

As you've seen above, an object of the class `gwaa.data-class` or `snp.data-class` is sub-settable in standard manner: `[i,j]`, where `i` is an index of a study subject and `j` is an index of a SNP. Importantly, `i` could be a list of indexes:

```
> vec <- which(srdta@phdata$age >= 65)
> vec

 [1]   64  122  186  206  207  286  385  386  492  514  525  536  545  565  613
[16]  632  649  673  701  779  799  981 1008 1131 1186 1223 1281 1383 1471 1489
[31] 1501 1565 1584 1673 1679 1782 1821 1832 1866 1891 1953 2081 2085 2140 2224
[46] 2268 2291 2384 2420 2453

> summary(srdta@gtdata[vec, 1:3])

     NoMeasured CallRate       Q.2 P.11 P.12 P.22    Pexact        Fmax
rs10         48     0.96 0.1354167   36   11    1 1.0000000  0.02131603
rs18         47     0.94 0.2765957   25   18    4 0.7245853  0.04298643
rs29         45     0.90 0.1555556   32   12    1 1.0000000 -0.01503759
          Plrt Chromosome
rs10 0.8843626          1
rs18 0.7697067          1
rs29 0.9188943          1
```

This shows summary of first three genotypes for people with age greater then or equal to 65 y.o. The same result may be achieved by sub-setting using a vector of logical values:

```
> vec <- (srdta@phdata$age >= 65)
> table(vec)

vec
FALSE  TRUE
 2450    50

> summary(srdta@gtdata[vec, 1:3])

     NoMeasured CallRate       Q.2 P.11 P.12 P.22    Pexact        Fmax
rs10         48     0.96 0.1354167   36   11    1 1.0000000  0.02131603
rs18         47     0.94 0.2765957   25   18    4 0.7245853  0.04298643
rs29         45     0.90 0.1555556   32   12    1 1.0000000 -0.01503759
          Plrt Chromosome
rs10 0.8843626          1
rs18 0.7697067          1
rs29 0.9188943          1
```

or a list with IDs of study subjects:

```
> vec1 <- srdta@gtdata@idnames[vec]
> vec1

 [1] "p64"   "p122"  "p186"  "p206"  "p207"  "p286"  "p385"  "p386"  "p492"
[10] "p514"  "p525"  "p536"  "p545"  "p565"  "p613"  "p632"  "p649"  "p673"
[19] "p701"  "p779"  "p799"  "p981"  "p1008" "p1131" "p1186" "p1223" "p1281"
[28] "p1383" "p1471" "p1489" "p1501" "p1565" "p1584" "p1673" "p1679" "p1782"
[37] "p1821" "p1832" "p1866" "p1891" "p1953" "p2081" "p2085" "p2140" "p2224"
[46] "p2268" "p2291" "p2384" "p2420" "p2453"

> summary(srdta@gtdata[vec1, 1:3])

     NoMeasured CallRate       Q.2 P.11 P.12 P.22    Pexact        Fmax
rs10         48     0.96 0.1354167   36   11    1 1.0000000  0.02131603
rs18         47     0.94 0.2765957   25   18    4 0.7245853  0.04298643
rs29         45     0.90 0.1555556   32   12    1 1.0000000 -0.01503759
          Plrt Chromosome
rs10 0.8843626          1
rs18 0.7697067          1
rs29 0.9188943          1
```

Let us explore the object returned by `summary` function when applied to `snp.data` class in more details:

```
> a <- summary(srdta@gtdata[vec1, 1:3])
> class(a)

[1] "data.frame"
```

Thus, the object returned is a `data.frame`. Therefore it should have dimensions and names:

```
> dim(a)

[1]  3 10

> names(a)

 [1] "NoMeasured" "CallRate"   "Q.2"        "P.11"       "P.12"
 [6] "P.22"       "Pexact"     "Fmax"       "Plrt"       "Chromosome"
```

Indeed, we derived 8 characteristics ("NoMeasured", "CallRate", "Q.2", "P.11", "P.12", "P.22", "Pexact", "Chromosome") for the first 3 SNPs.

Exercise 27 *Test if Hardy-Weinberg equilibrium holds for the first 10 SNPs*

1. *Total sample*
2. *In cases (`bt` is 1)*
3. *In controls (`bt` is 0)*

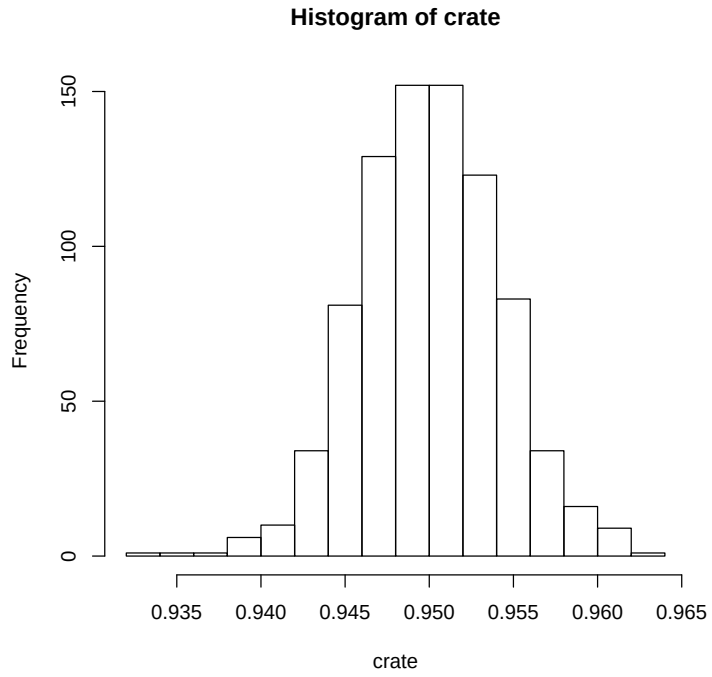

Figure 7: Histogram of the call rate

Let us analyse the distribution of call rate in the whole study. For this, we first need to obtain the vector of call rates:

```
> sumgt <- summary(srdta@gtdata)
> crate <- sumgt[, "CallRate"]
```

This vector may be presented by a histogram

```
> hist(crate)
```

which shows that most SNPs have call rate between 93 and 97% (figure 7).

As next step, you would like to produce a summary table, showing how many markers had call rate lower than, say, 93%, between 93 and 95%, between 95 and 99% and more than 99%. You can use `catable()` command for that:

```
> catable(crate, c(0.93, 0.95, 0.99))
```

```
     X<=0.93 0.93<X<=0.95 0.95<X<=0.99 X>0.99
No         0      415.000      418.000      0
Prop       0        0.498        0.502      0
```

Similar procedure may be applied to see deviation from HWE:

```
> hwp <- sumgt[, "Pexact"]
> catable(hwp, c((0.05/srdta@gtdata@nsnps), 0.01, 0.05, 0.1))
```

```
     X<=6.00240096038415e-05 6.00240096038415e-05<X<=0.01 0.01<X<=0.05
No                     2.000                        7.000       23.000
Prop                   0.002                        0.008        0.028
     0.05<X<=0.1   X>0.1
No        31.000 770.000
Prop       0.037   0.924
```

The first cut-off category will detect SNPs which are deviating from HWE at the Bonferroni-corrected P-level.

However, for these data it will make more sense to table cumulative distribution:

```
> catable(hwp, c((0.05/srdta@gtdata@nsnps), 0.01, 0.05, 0.1), cum = T)

     X<=6.00240096038415e-05 X<=0.01 X<=0.05 X<=0.1 all X
No                     2.000   9.000  32.000 63.000   833
Prop                   0.002   0.011   0.038  0.076     1
```

If you would like to investigate the minor allele frequency (MAF) distribution, the same logic would apply. First, derive MAF with

```
> afr <- sumgt[, "Q.2"]
> maf <- pmin(afr, (1 - afr))
```

Next, generate histograms for frequency and MAF:

```
> par(mfcol = c(2, 1))
> hist(afr)
> hist(maf)
```

(shown at the figure 8) and then generate table describing frequency distribution:

```
> catable(afr, c(0.01, 0.05, 0.1, 0.2, 0.5, 0.8, 0.9, 0.95, 0.99))

     X<=0.01 0.01<X<=0.05 0.05<X<=0.1 0.1<X<=0.2 0.2<X<=0.5 0.5<X<=0.8
No    22.000       53.000      99.000    132.000    313.000    187.000
Prop   0.026        0.064       0.119      0.158      0.376      0.224
     0.8<X<=0.9 0.9<X<=0.95 0.95<X<=0.99 X>0.99
No       18.000        8.00        1.000      0
Prop      0.022        0.01        0.001      0

> catable(maf, c(0, 0.01, 0.05, 0.1, 0.2), cum = T)

     X<=0 X<=0.01 X<=0.05 X<=0.1 X<=0.2 all X
No      0  22.000  76.000 183.00  333.0   833
Prop    0   0.026   0.091   0.22    0.4     1
```

Note that we used "0" as the first category – this will give you the number of monomorhic SNPs which we recommend to exclude from analysis.

Other function, `perid.summary`, produces summary SNP statistics per person. Let us try producing this summary for the first 10 people:

```
> perid.summary(srdta[1:10, ])
```

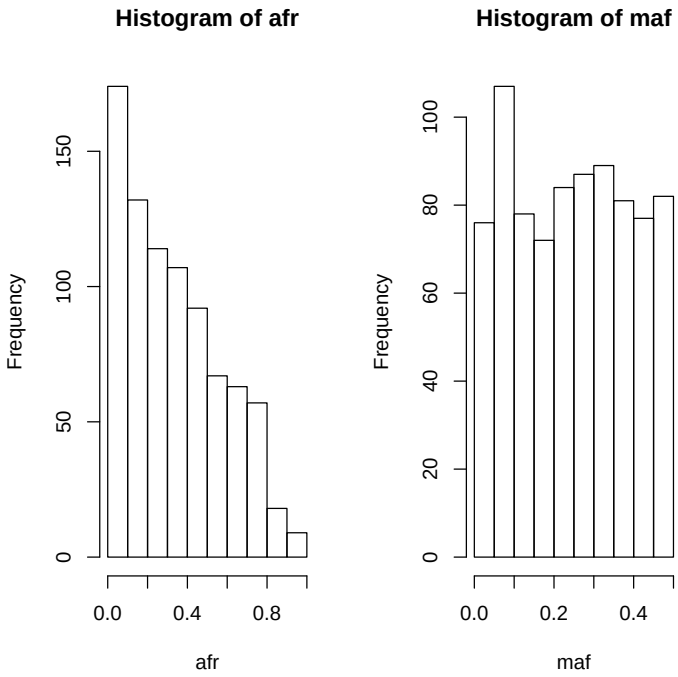

Figure 8: Histogram of the call rate

```
    NoMeasured    CallPP       Het
p1         790 0.9483794 0.2012658
p2         792 0.9507803 0.2525253
p3         783 0.9399760 0.3793103
p4         789 0.9471789 0.3929024
p5         790 0.9483794 0.3341772
p6         787 0.9447779 0.2337992
p7         794 0.9531813 0.3690176
p8         793 0.9519808 0.2976040
p9         788 0.9459784 0.3324873
p10        797 0.9567827 0.3412798
```

This table lists the number of genotypes scored for the person, call rate, and heterozygosity. The outliers who have increased average heterozygosity may be suggestive of contaminated DNA samples.

Let us analyse the distribution of heterozygosity:

```
> het <- perid.summary(srdta)$Het
> mean(het)

[1] 0.3309457

> catable(het, c(0.1, 0.25, 0.3, 0.35, 0.5))

     X<=0.1 0.1<X<=0.25 0.25<X<=0.3 0.3<X<=0.35 0.35<X<=0.5 X>0.5
No    7.000      73.000     339.000    1281.000      800.00     0
Prop  0.003       0.029       0.136       0.512        0.32     0

> plot(het)
```

The resulting histogram is presented in figure 9. It is easy to see that few people have very low heterozygosity, but there are no outliers with extremely high values.

In this section, we covered low-level functions `summary` and `perid.summary`. Base on these, an upper-level genetic data quality control function, `check.marker`, is based. That function will be covered in the next section.

Summary:

- When `summary()` function is applied to an object of `snp.data-class`, it return summary statistics for SNPs, including exact test for Hardy-Weinberg equilibrium.
- When `perid.summary()` function is applied to an object of `snp.data-class`, it return per-person summary statistics, including the call rate within this person and its' heterozygosity.

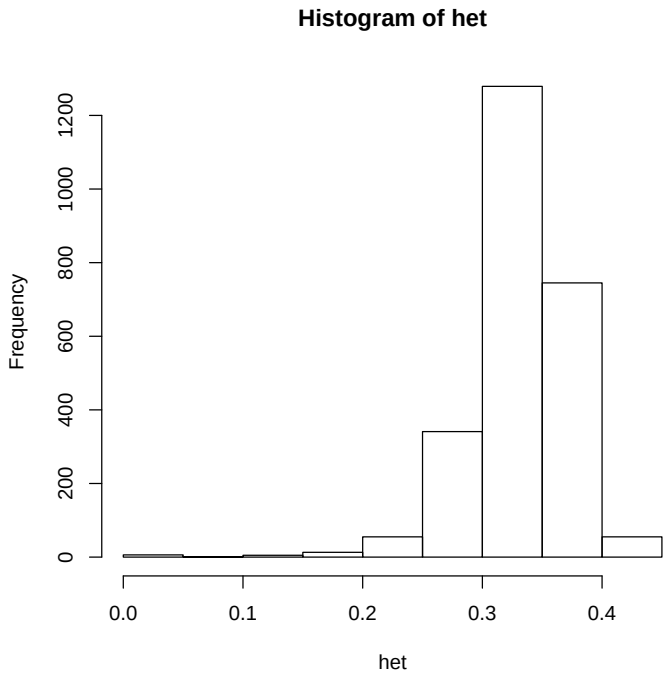

Figure 9: Histogram of heterozygosity

Exercise 28 *Characterise the distribution of call rates within study subjects and produce a histogram. How many people have call rate below 93%?*

5 Genome-wide association analysis

In the first parts of this section you will be guided through a GWA analysis of a small data set. In the last part you will investigate a larger data set by yourself, do a verification study and will answer the questions. All data sets used assume a study in a relatively homogeneous population. Try to finish the first part in the morning and the second part in the afternoon.

Though only few thousands of markers located at four small chromosomes are used in the scan, we still going to call it Genome-Wide (GW), as the amount of data we will use is approaches the amount to be expected in a real experiment. However, because the regions are small, and the LD between SNPs is high, some specific features (e.g. relatively high residual inflation, which occurs because large proportion of SNPs are in LD with the reuly associated ones) are specific features of this data set, which are not observed in true GWA studies.

Start R and load `GenABEL` library by typing

```
> library(GenABEL)
```

and load the data which we will use in this section by

```
> data(ge03d2ex)
```

Investigate the objects loaded by command

```
> ls()
```

```
[1] "ge03d2ex"
```

The `ge03d2ex` is an object of the class `gwaa.data`:

```
> class(ge03d2ex)
```

```
[1] "gwaa.data"
attr(,"package")
[1] "GenABEL"
```

To check what are the names of variables in the phenotypic data frame, use

```
> names(ge03d2ex@phdata)
```

```
[1] "id"     "sex"    "age"    "dm2"    "height" "weight" "diet"   "bmi"
```

We can attach this data frame to the `R` search path by

```
> attach(ge03d2ex@phdata)
```

5.1 Data descriptives and first round of GWA analysis

Let us investigate what are the traits presented in the data frame loaded and what are the characteristics of the distribution by using specific `GenABEL` function `descriptive.trait`:

```
> descriptives.trait(ge03d2ex)
```

```
        No    Mean     SD
id     136      NA     NA
sex    136   0.529  0.501
age    136  49.069 12.926
dm2    136   0.632  0.484
height 135 169.440  9.814
weight 135  87.397 25.510
diet   136   0.059  0.236
bmi    135  30.301  8.082
```

You can see that phenotypic frame contains the data on 136 people; the data on sex, age, height, weight, diet and body mass index (BMI) are available. Our trait of interest is `dm2` (type 2 diabetes). Note that every single piece of information in this data set is simulated; however, we tried to keep our simulations in a way we think the control of T2D may work.

You can produce a summary for cases and controls separately and compare distributions of the traits by

```
> descriptives.trait(ge03d2ex, by = dm2)
```

```
       No(by.var=1)    Mean     SD No(by.var=0)    Mean     SD   Ptt   Pkw
id               86      NA     NA           50      NA     NA    NA    NA
sex              86   0.593  0.494           50   0.420  0.499 0.053 0.052
age              86  50.250 12.206           50  47.038 13.971 0.179 0.205
dm2              86      NA     NA           50      NA     NA    NA    NA
height           86 170.448 10.362           49 167.671  8.586 0.097 0.141
weight           86  93.587 27.337           49  76.534 17.441 0.000 0.000
diet             86   0.058  0.235           50   0.060  0.240 0.965 0.965
bmi              86  32.008  8.441           49  27.304  6.463 0.000 0.001
       Pexact
id         NA
sex     0.074
age        NA
dm2        NA
height     NA
weight     NA
diet    1.000
bmi        NA
```

Here, the `by` argument specifies the grouping variable. You can see that cases and controls are different in weight, which is expected, as T2D is associated with obesity.

Similarly, you can produce grand GW descriptives of the marker data by using

```
> descriptives.marker(ge03d2ex)

$`Minor allele frequency distribution`
     X<=0.01 0.01<X<=0.05 0.05<X<=0.1 0.1<X<=0.2    X>0.2
No   146.000      684.000     711.000    904.000 1555.000
Prop   0.036        0.171       0.178      0.226    0.389

$`Cumulative distr. of number of SNPs out of HWE, at different alpha`
     X<=1e-04 X<=0.001 X<=0.01 X<=0.05 all X
No     46.000   71.000 125.000 275.000  4000
Prop    0.011    0.018   0.031   0.069     1

$`Distribution of porportion of successful genotypes (per person)`
     X<=0.9 0.9<X<=0.95 0.95<X<=0.98 0.98<X<=0.99 X>0.99
No    1.000           0            0      135.000      0
Prop  0.007           0            0        0.993      0

$`Distribution of porportion of successful genotypes (per SNP)`
     X<=0.9 0.9<X<=0.95 0.95<X<=0.98 0.98<X<=0.99   X>0.99
No   37.000       6.000      996.000     1177.000 1784.000
Prop  0.009       0.002        0.249        0.294    0.446

$`Mean heterozygosity for a SNP`
[1] 0.2582298

$`Standard deviation of the mean heterozygosity for a SNP`
[1] 0.1592255

$`Mean heterozygosity for a person`
[1] 0.2476507

$`Standard deviation of mean heterozygosity for a person`
[1] 0.04291038
```

It is of note that we can see inflation of the proportion of the tests for HWE at particular threshold, as compared to the expected. This may indicate poor genotyping quality and/or genetic stratification.

We can test the GW marker characteristics in controls by

```
> descriptives.marker(ge03d2ex, ids = (dm2 == 0))

$`Minor allele frequency distribution`
     X<=0.01 0.01<X<=0.05 0.05<X<=0.1 0.1<X<=0.2    X>0.2
No   233.000      676.000     671.000    898.000 1522.000
Prop   0.058        0.169       0.168      0.225    0.381

$`Cumulative distr. of number of SNPs out of HWE, at different alpha`
     X<=1e-04 X<=0.001 X<=0.01 X<=0.05 all X
No          0    3.000  14.000  98.000  4000
Prop        0    0.001   0.003   0.025     1
```

```
$`Distribution of porportion of successful genotypes (per person)`
     X<=0.9 0.9<X<=0.95 0.95<X<=0.98 0.98<X<=0.99 X>0.99
No        0           0            0           50      0
Prop      0           0            0            1      0

$`Distribution of porportion of successful genotypes (per SNP)`
     X<=0.9 0.9<X<=0.95 0.95<X<=0.98 0.98<X<=0.99   X>0.99
No   37.000      49.000     1523.000            0 2391.000
Prop  0.009       0.012        0.381            0    0.598

$`Mean heterozygosity for a SNP`
[1] 0.2555009

$`Standard deviation of the mean heterozygosity for a SNP`
[1] 0.1618707

$`Mean heterozygosity for a person`
[1] 0.2525720

$`Standard deviation of mean heterozygosity for a person`
[1] 0.04714886
```

Apparently, HWE distribution holds better in controls than in the total sample.

Let us check whether there are indications that deviation from HWE is due to cases. At this stage we are only interested in HWE distribution table, and therefore will ask to report only table two:

```
> descriptives.marker(ge03d2ex, ids = (dm2 == 1))[2]

$`Cumulative distr. of number of SNPs out of HWE, at different alpha`
     X<=1e-04 X<=0.001 X<=0.01 X<=0.05 all X
No     45.000    79.00 136.000 268.000  4000
Prop    0.011     0.02   0.034   0.067     1
```

It seems that indeed excessive number of markers are out of HWE in cases. If no laboratory procedure (e.g. DNA extraction, genotyping, calling) were done for cases and controls separately, this may indicate possible heterogeneity specific for cases.

It may be interesting to plot a $\chi^2 - \chi^2$ plot contrasting observed and expected distributions for the test for HWE in cases. First, we need to compute summary SNP statistics by

```
> s <- summary(ge03d2ex@gtdata[(dm2 == 1), ])
```

Note the you have produced the summary for the `gtdata` slot of `ge03d2ex`; this is the slot which actually contain all genetic data in special compressed format.

You can see first 10 elements of this very long table by

```
> s[1:10, ]
```

```
          NoMeasured  CallRate        Q.2 P.11 P.12 P.22      Pexact
rs7435137         84 0.9767442 0.52380952   17   46   21 0.510978370
rs7725697         85 0.9883721 0.01176471   83    2    0 1.000000000
rs664063          86 1.0000000 0.08720930   71   15    0 1.000000000
rs4670072         60 0.6976744 0.11666667   53    0    7 0.001701645
rs546570          84 0.9767442 0.89880952    1   15   68 1.000000000
rs7908680         83 0.9651163 0.03012048   78    5    0 1.000000000
rs166732          83 0.9651163 0.04216867   76    7    0 1.000000000
rs4257079         86 1.0000000 0.07558140   73   13    0 1.000000000
rs5150804         84 0.9767442 0.39880952   31   39   14 0.820496827
rs3508821         83 0.9651163 0.20481928   52   28    3 1.000000000
                 Fmax         Plrt Chromosome
rs7435137 -0.09772727 0.3699602726          1
rs7725697 -0.01190476 0.8773691192          3
rs664063  -0.09554140 0.2308999066          2
rs4670072  1.00000000 0.0002510899          X
rs546570   0.01830931 0.8693645189          2
rs7908680 -0.03105590 0.6935168932          1
rs166732  -0.04402516 0.5787401988          1
rs4257079 -0.08176101 0.3022863648          1
rs5150804  0.03177183 0.7710793180          2
rs3508821 -0.03565062 0.7419587406          2
```

Note that the column before the last provides P-exact we need. We can extract these to a separate vector by

```
> pexcas <- s[, "Pexact"]
```

and produce the $\chi^2 - \chi^2$ plot and estimate inflation factor by command `estlambda()`, which operates with a vector of P-values or χ^2s:

```
> estlambda(pexcas)

$estimate
[1] 1.068184

$se
[1] 0.02614764
```

By default, this function also produces a $\chi^2 - \chi^2$ plot, at which you can see some extreme deviation of observed from expected. The resulting plot (figure 10) shows extreme deviation for high values of the test. Looking at the λ estimate, we indeed see inflation of the test statistics.

You can repeat this test for the controls, if time permits.

The 'se' produced by `estlambda` *can not* be used to test if inflation is significant and make conclusions about presence of stratification.

Let us first try do GWA scan using raw (before quality control) data. We will use the score test, as implemented in the `qtscore()` funcrtion of `GenABEL` for testing:

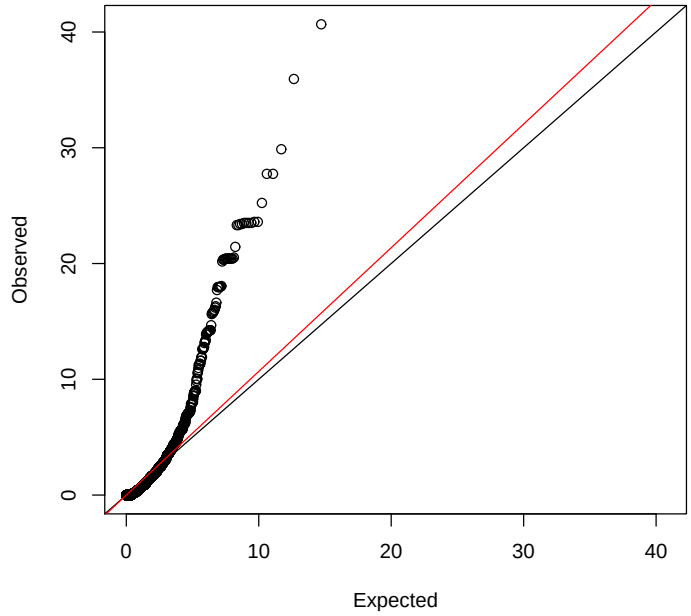

Figure 10: $\chi^2 - \chi^2$ plot for the exact test for HWE. Black line of slope 1: expected under no inflation; Red line: fitted slope.

```
> an0 <- qtscore(dm2, ge03d2ex, trait = "binomial")
```

The first argument used describes the model; here it is rather simple — the affection status, `dm2`, is supposed to depend on SNP genotype only.

You can see what objects are returned by this function by using

```
> names(an0)
```

```
 [1] "chi2.1df"   "chi2.2df"   "P1df"       "P2df"       "Pc1df"
 [6] "Pc2df"      "lambda"     "effB"       "effAB"      "effBB"
[11] "N"          "snpnames"   "idnames"    "map"        "chromosome"
[16] "formula"    "family"
```

Here, `P1df`, `P2df` and `Pc1df` are most interesting; the first two are vectors of 1 and 2 d.f. P-values obtained in the GWA analysis, the last one is 1 d.f. P-value corrected for inflation factor λ (which is presented in `lambda` object).

Let us see if there is evidence for the inflation of the test statistics

```
> an0$lambda
```

```
$estimate
[1] 1.047846

$iz0
```

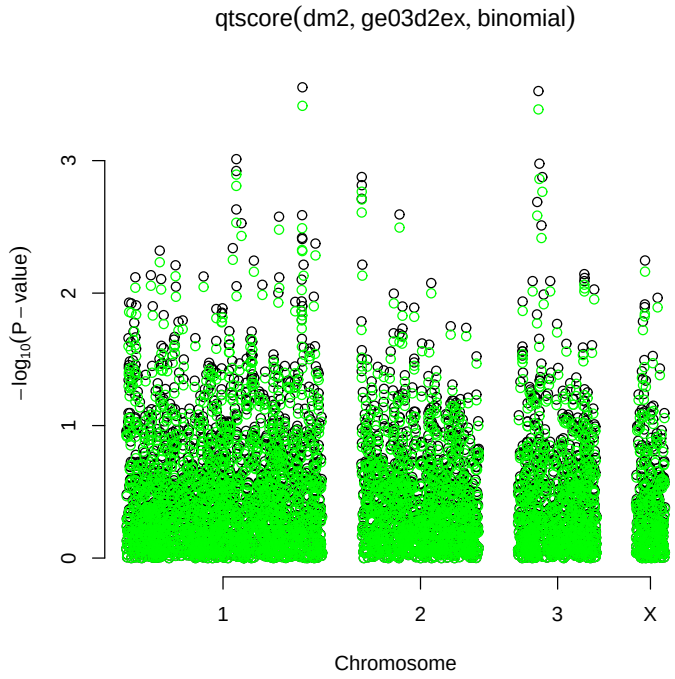

Figure 11: $-log_{10}(P-value)$ from the genome scan before QC procedure. Raw analysis: black; corrected analysis: green

```
[1] 1.002987

$iz2
[1] 0.8084199
```

The estimate of λ is 1.05, suggesting inflation of the test and some degree of stratification.

We can plot the results of analysis by

```
> plot(an0)
```

The resulting plot is presented in the figure 11. By default, $-log_{10}(P-value)$ on 1 d.f. are presented; see help to figure out how this behaviour can be changed.

We can also add the corrected P-values to the plot with

```
> add.plot(an0, df = "Pc1df", col = "green")
```

You can also generate a descriptive table for the "top" (as ranked by P-value) results by

```
> descriptives.scan(an0)
```

```
          Chromosome Position   N     effB     P1df    Pc1df    effAB     effBB
rs1719133          1  4495479 136 0.337293 0.000280 0.000386 0.400424  0.000000
```

```
rs2975760          3 10518480 134 3.803800 0.000298 0.000411 3.454545 10.000000
rs7418878          1  2808520 136 3.081231 0.000974 0.001274 3.605128  4.871795
rs5308595          3 10543128 133 3.982550 0.001054 0.001375 3.317143       Inf
rs4804634          1  2807417 132 0.434115 0.001197 0.001552 0.524064  0.173913
rs3224311          2  6009769 135 3.158317 0.001329 0.001716 3.415179  4.250000
rs26325            3 10617781 135 0.097428 0.001331 0.001719 0.109772        NA
rs8835506          2  6010852 132 3.177208 0.001532 0.001966 3.490385  4.125000
rs3925525          2  6008501 135 2.984169 0.001940 0.002464 3.238095  4.121212
rs2521089          3 10487652 135 2.502395 0.002052 0.002601 2.571770  4.772727
              P2df
rs1719133 0.000633
rs2975760 0.001143
rs7418878 0.002264
rs5308595 0.004593
rs4804634 0.003696
rs3224311 0.002941
rs26325   0.001331
rs8835506 0.003162
rs3925525 0.004555
rs2521089 0.006966
```

Here you see top 10 results, sorted by P-value with 1 d.f. If you want to sort by the corrected P-value, you can use `descriptives.scan(an0,sort="Pc1df")`; to see more then 10 (e.g. 25) top results, use `descriptives.scan(an0,top=25)`. You can combine all these options. Large part of results reports NA as effect estimates and 9.99 as *P-value* for 2 d.f. test – for these markers only two out of three possible genotypes were observed, and consequently 2 d.f. test could not be performed.

Now let us apply `emp.qtscore()` function, which computes empirical GW (or experiment-wise) significance

```
> an0.e <- emp.qtscore(dm2, ge03d2ex)

100%

> descriptives.scan(an0.e, sort = "Pc1df")

          Chromosome Position   N      effB P1df Pc1df     effAB     effBB P2df
rs1719133          1  4495479 136 -0.265206 0.43  0.53 -0.208088 -0.737500    1
rs2975760          3 10518480 134  0.234065 0.44  0.55  0.275510  0.409091    1
rs7418878          1  2808520 136  0.208910 0.85  0.93  0.280741  0.326840    1
rs5308595          3 10543128 133  0.244552 0.88  0.95  0.256483  0.462366    1
rs4804634          1  2807417 132 -0.205045 0.93  0.96 -0.119383 -0.384524    1
rs3224311          2  6009769 135  0.213363 0.94  0.97  0.277863  0.315152    1
rs26325            3 10617781 135 -0.487537 0.94  0.97 -0.487537        NA    1
rs8835506          2  6010852 132  0.211200 0.96  0.99  0.279622  0.307692    1
rs7435137          1  4259040 134  0.073451 1.00  1.00  0.116637  0.146169    1
rs7725697          3 10806991 135  0.375940 1.00  1.00  0.375940        NA    1
```

None of the SNPs hits GW significance. If, actually, any did pass the threshold, we could not trust the results, because the distribution of the HWE test

and presence of inflation factor for the association test statistics suggest that the data may contain multiple errors (indeed they do). Therefore before association analysis we need to do rigorous Quality Control (QC).

Note that at certain SNP, corrected *P-values* become equal to 1 – at this point order is arbitrary because sorting could not be done.

Summary:

- The `descriptives` family of functions was developed to facilitate production of tables which can be directly used in a manuscript — it is possible to save the output as a file, which can be open by Excel or Word. See e.g. `help(descriptives.trait)` for details.
- The inflation of test statistics compared to null (1 d.f.) may be estimated with `estlambda` function.

5.2 Genetic data QC

The major genetic data QC function of `GenABEL` is `check.marker()`. We will try to run it; the output is rather self-explaining. As it was detailed at the lecture, in the first round of the QC we do not want to check for HWE. This can be achieved by setting HWE P-value selection threshold to zero (`p.level=0`):

```
> qc1 <- check.marker(ge03d2ex, p.level = 0)

Excluding people/markers with extremely low call rate...
4000 markers and 136 people in total
0 people excluded because of call rate < 0.1
6 markers excluded because of call rate < 0.1
Passed: 3994 markers and 136 people

Running sex chromosome checks...
197 heterozygous X-linked male genotypes found
1 X-linked markers are likely to be autosomal (odds > 1000 )
2 male are likely to be female (odds > 1000 )
0 female are likely to be male (odds > 1000 )
If these people/markers are removed, 0 heterozygous male genotypes are left
Passed: 3993 markers and 134 people

no X/Y/mtDNA-errors to fix


RUN 1
3993 markers and 134 people in total
304 (7.613323%) markers excluded as having low (<1.865672%) minor allele frequency
36 (0.9015778%) markers excluded because of low (<95%) call rate
0 (0%) markers excluded because they are out of HWE (P <0)
1 (0.7462687%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2747262 (s.e. 0.03721277)
```

```
3 (2.238806%) people excluded because too high autosomal heterozygosity (FDR <1%)
Excluded people had HET >= 0.4856887
Mean IBS is 0.7734414 (s.e. 0.02122081), as based on 2000 autosomal markers
1 (0.7462687%) people excluded because of too high IBS (>=0.95)
In total, 3653 (91.4851%) markers passed all criteria
In total, 130 (97.01493%) people passed all criteria

RUN 2
3653 markers and 130 people in total
69 (1.888858%) markers excluded as having low (<1.923077%) minor allele frequency
0 (0%) markers excluded because of low (<95%) call rate
0 (0%) markers excluded because they are out of HWE (P <0)
0 (0%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2745297 (s.e. 0.01722736)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7715355 (s.e. 0.01741012), as based on 2000 autosomal markers
1 (0.7692308%) people excluded because of too high IBS (>=0.95)
In total, 3584 (98.11114%) markers passed all criteria
In total, 130 (100%) people passed all criteria

RUN 3
3584 markers and 130 people in total
0 (0%) markers excluded as having low (<1.923077%) minor allele frequency
0 (0%) markers excluded because of low (<95%) call rate
0 (0%) markers excluded because they are out of HWE (P <0)
0 (0%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2745297 (s.e. 0.01722736)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7707034 (s.e. 0.01772056), as based on 2000 autosomal markers
1 (0.7692308%) people excluded because of too high IBS (>=0.95)
In total, 3584 (100%) markers passed all criteria
In total, 130 (100%) people passed all criteria
```

The computation of all pairwise proportion of alleles identical-by-state (IBS) by `ibs()` function, which is also called by `check.markers()` may take quite some time, which is proportional to the square of the number of subjects. This is not a problem with the small number of people we use for this example or when modern computers are used. However, the computers in the Nihes computer room are very old. Therefore be prepared to wait for long time when you will do a self-exercise with 1,000 people.

From the output you can see that QC starts with checking the data for SNPs and people with extremely low call rate. Six markers are excluded from further analysis due to very low call rate. Next, X-chromosomal errors are identified. The function finds out that all errors (heterozygous male X-genotypes) are due to two people with wrong sex assigned and one marker, which looks like an autosomal one. This actually could be a marker from pseudoautosomal region, which should have been arranged as a separate ”autosome”.

Then, the procedure finds the markers with low call rate (≤ 0.95) across people, markers with low MAF (by default, low MAF is defined as less than few copies of the rare allele, see help for details); people with low call rate (≤ 0.95) across SNPs, people with extreme heterozygosity (at FDR 0.01) and these who have GW IBS ≥ 0.95. These default parameters may be changed if you wish (consult help).

Because some of the people fail to pass the tests, the data set is not guaranteed to be really "clean" after single iteration, e.g. some marker may not pass the call threshold after we exclude few informative (but apparently wrong) people. Therefore the QC is repeated iteratively until no further errors are found.

You can generate short summary of QC by marker and by person through

```
> summary(qc1)

$`Per-SNP fails statistics`
          NoCall NoMAF NoHWE Redundant Xsnpfail
NoCall        42     0     0         0        0
NoMAF         NA   373     0         0        0
NoHWE         NA    NA     0         0        0
Redundant     NA    NA    NA         0        0
Xsnpfail      NA    NA    NA        NA        1

$`Per-person fails statistics`
         IDnoCall HetFail IBSFail isfemale ismale isXXY
IDnoCall        1       0       0        0      0     0
HetFail        NA       3       0        0      0     0
IBSFail        NA      NA       1        0      0     0
isfemale       NA      NA      NA        2      0     0
ismale         NA      NA      NA       NA      0     0
isXXY          NA      NA      NA       NA     NA     0
```

Note that the original data, `ge03d2ex`, are not modified during the procedure; rather, `check.markers()` generate a list of markers and people which pass or do not pass certain QC criteria. The objects returned by `check.markers()` are:

```
> names(qc1)

 [1] "nofreq"   "nocall"   "nohwe"    "Xmrkfail" "hetfail"  "idnocall"
 [7] "ibsfail"  "isfemale" "ismale"   "snpok"    "idok"     "call"
```

The element `idok` provides the list of people who passed all QC criteria, and `snpok` provides the list of SNPs which passed all criteria. You can easily generate a new data set, which will consist only of these people and markers by

```
> data1 <- ge03d2ex[qc1$idok, qc1$snpok]
```

If there are any residual sporadic X-errors (male heterozygosity), these can be fixed (set to NA) by

```
> data1 <- Xfix(data1)

no X/Y/mtDNA-errors to fix
```

Applying this function does not make any difference for the example data set, but you will need to use it for the bigger data set.

At this point, we are ready to work with the new, cleaned, data set `data1`. However, if we try

```
> length(dm2)

[1] 136
```

we can see that the original phenotypic data are attached to the search path (there are only 130 people left in the 'clean' data set). Therefore we need to detach the data by

```
> detach(ge03d2ex@phdata)
```

and attach new data by

```
> attach(data1@phdata)
```

At this stage, let us check if the first round of QC solves the problem of inflated test for HWE, which may be the case if this inflation is due to genotypic errors we managed to eliminate:

```
> descriptives.marker(data1)[2]

$`Cumulative distr. of number of SNPs out of HWE, at different alpha`
     X<=1e-04 X<=0.001 X<=0.01 X<=0.05 all X
No     43.000   66.000 124.000 253.000  3584
Prop    0.012    0.018   0.035   0.071     1

> descriptives.marker(data1[dm2 == 1])[2]

$`Cumulative distr. of number of SNPs out of HWE, at different alpha`
     X<=1e-04 X<=0.001 X<=0.01 X<=0.05 all X
No     46.000    72.00 125.000 235.000  3584
Prop    0.013     0.02   0.035   0.066     1

> estlambda(summary(data1@gtdata[dm2 == 1, ])[, "Pexact"])

$estimate
[1] 1.102509

$se
[1] 0.02784137
```

Apparently, the distribution (figure 12) looks better (note the scale difference between the graphs), but the test statistics is still inflated.

5.3 Finding genetic sub-structure

Now, we are ready for the second round of QC, detection of genetic outliers which may contaminate our data. We will detect genetic outliers using a technique, which resembles the one suggested by Price at al.

As a first step, we will compute a matrix of genomic kinship between all pairs of people, using only autosomal markers by

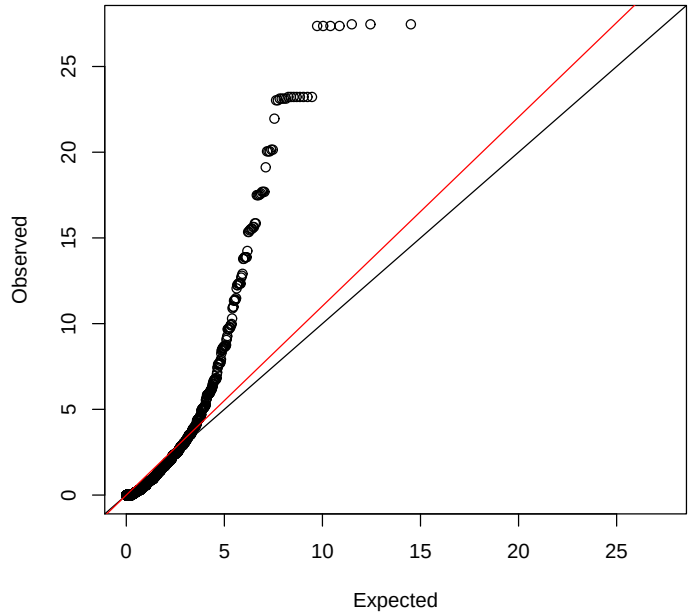

Figure 12: $\chi^2 - \chi^2$ plot for the exact test for HWE. Black line of slope 1: expected under no inflation; Red line: fitted slope.

```
> data1.gkin <- ibs(data1[, data1@gtdata@chromosome != "X"], weight = "freq")
```

You can see the 5x5 upper left sub-matrix by

```
> data1.gkin[1:5, 1:5]
```

```
             id199         id300         id403         id415       id666
id199  0.497044383 3265.00000000 3264.00000000 3252.00000000 3268.000000
id300 -0.012711493    0.49531467 3271.00000000 3260.00000000 3274.000000
id403 -0.012653681   -0.01258717    0.51882653 3258.00000000 3273.000000
id415 -0.002906311    0.01344947   -0.02541456    0.56778632 3262.000000
id666 -0.019453258   -0.02094126    0.02115588   -0.02037597    0.568936
```

This step may take few minutes on large data sets or when using old computers!

The numbers below the diagonal show genomic kinship (IBD), the numbers on the diagonal correspond to 0.5 plus the genomic homozigosity, and the numbers above the diagonal tell how many SNPs were typed successfully for both subjects (thus the IBD estimate is derived using this number of SNPs).

Second, we transform this matrix to a distance matrix using standard `R` command

```
> data1.dist <- as.dist(0.5 - data1.gkin)
```

Finally, we perform Classical Multidimensional Scaling by

```
> data1.mds <- cmdscale(data1.dist)
```

by default, the first two principal components are computed and returned.

This may take few minutes on large data sets or when using old computers!

We can present the results graphically by

```
> plot(data1.mds)
```

The resulting plot is presented in figure 13. Each point on the plot corresponds to a person, and the 2D distances between points were fitted to be as close as possible to these presented in the original IBS matrix. You can see that study subjects clearly cluster in two groups.

You can identify the points belonging to clusters by

```
> km <- kmeans(data1.mds, centers = 2, nstart = 1000)
> cl1 <- names(which(km$cluster == 1))
> cl2 <- names(which(km$cluster == 2))
> cl1
```

```
  [1] "id199"  "id300"  "id403"  "id415"  "id666"  "id689"  "id765"  "id830"
  [9] "id908"  "id980"  "id994"  "id1193" "id1423" "id1505" "id1737" "id1827"
 [17] "id1841" "id2068" "id2094" "id2115" "id2151" "id2317" "id2618" "id2842"
 [25] "id2894" "id2985" "id3354" "id3368" "id3641" "id3831" "id3983" "id4097"
 [33] "id4328" "id4380" "id4395" "id4512" "id4552" "id4710" "id4717" "id4883"
 [41] "id4904" "id4934" "id4961" "id5014" "id5078" "id5274" "id5275" "id5454"
 [49] "id5853" "id5926" "id5969" "id6237" "id6278" "id6352" "id6501" "id6554"
 [57] "id6663" "id6723" "id7499" "id7514" "id7541" "id7598" "id7623" "id7949"
 [65] "id8059" "id8128" "id8281" "id8370" "id8400" "id8433" "id8772" "id8880"
 [73] "id8890" "id8957" "id8996" "id9082" "id9901" "id9930" "id1857" "id2528"
 [81] "id4862" "id9184" "id5677" "id6407" "id5472" "id2135" "id8545" "id4333"
 [89] "id1670" "id1536" "id6917" "id6424" "id3917" "id9628" "id9635" "id4729"
 [97] "id5190" "id6399" "id6062" "id620"  "id1116" "id6486" "id41"   "id677"
[105] "id4947" "id9749" "id6428" "id7488" "id5949" "id2924" "id5783" "id4096"
[113] "id903"  "id9049" "id185"  "id1002" "id362"  "id9014" "id5044" "id2749"
[121] "id5437" "id2286" "id4743" "id4185" "id8330" "id6934"
```

```
> cl2
```

```
[1] "id2097" "id6954" "id2136" "id858"
```

Four outliers are presented in the smaller cluster.

Now you will need to use the BIGGER cluster for to select study subjects. Whether this will be `cl1` or `cl2` in you case, is totally random.

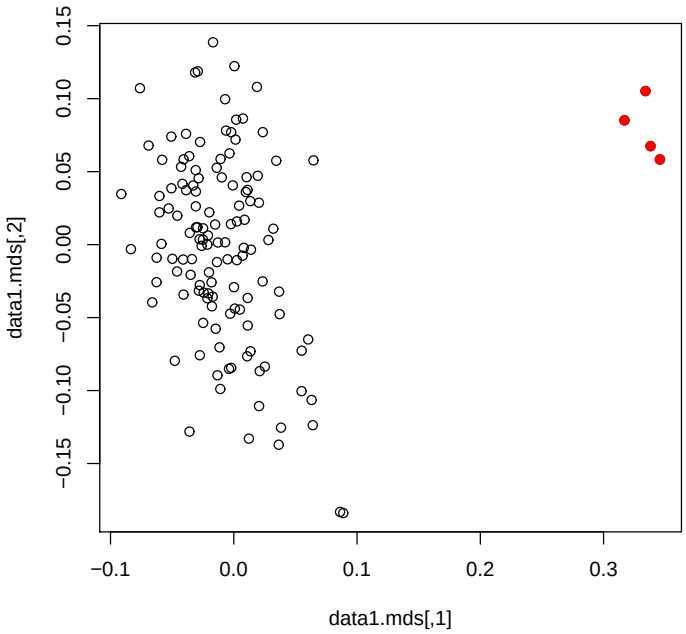

Figure 13: Mapping samples on the space of the first two Principle Components resulting from analysis of genomic kinship. Red dots identify genetic outliers

We can form a data set which is free from outliers by using only people from the bigger cluster:

```
> data2 <- data1[cl2, ]
```

After we dropped the outliers, we need to repeat QC using `check.markers()`. At this stage, we want to allow for HWE checks (we will use only controls and exclude markers with FDR ≤ 0.2):

```
> qc2 <- check.marker(data2, hweids = (data2@phdata$dm2 == 0),
+     fdr = 0.2)

Excluding people/markers with extremely low call rate...
3584 markers and 126 people in total
0 people excluded because of call rate < 0.1
0 markers excluded because of call rate < 0.1
Passed: 3584 markers and 126 people

Running sex chromosome checks...
0 heterozygous X-linked male genotypes found
0 X-linked markers are likely to be autosomal (odds > 1000 )
0 male are likely to be female (odds > 1000 )
0 female are likely to be male (odds > 1000 )
If these people/markers are removed, 0 heterozygous male genotypes are left
```

```
Passed: 3584 markers and 126 people

no X/Y/mtDNA-errors to fix


RUN 1
3584 markers and 126 people in total
39 (1.088170%) markers excluded as having low (<1.984127%) minor allele frequency
0 (0%) markers excluded because of low (<95%) call rate
0 (0%) markers excluded because they are out of HWE (FDR <0.2)
0 (0%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2776518 (s.e. 0.01673776)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7717277 (s.e. 0.01283029), as based on 2000 autosomal markers
1 (0.7936508%) people excluded because of too high IBS (>=0.95)
In total, 3545 (98.91183%) markers passed all criteria
In total, 126 (100%) people passed all criteria

RUN 2
3545 markers and 126 people in total
0 (0%) markers excluded as having low (<1.984127%) minor allele frequency
0 (0%) markers excluded because of low (<95%) call rate
0 (0%) markers excluded because they are out of HWE (FDR <0.2)
0 (0%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2776518 (s.e. 0.01673776)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7696476 (s.e. 0.01287892), as based on 2000 autosomal markers
1 (0.7936508%) people excluded because of too high IBS (>=0.95)
In total, 3545 (100%) markers passed all criteria
In total, 126 (100%) people passed all criteria

> summary(qc2)

$`Per-SNP fails statistics`
          NoCall NoMAF NoHWE Redundant Xsnpfail
NoCall         0     0     0         0        0
NoMAF         NA    39     0         0        0
NoHWE         NA    NA     0         0        0
Redundant     NA    NA    NA         0        0
Xsnpfail      NA    NA    NA        NA        0

$`Per-person fails statistics`
          IDnoCall HetFail IBSFail isfemale ismale isXXY
IDnoCall         0       0       0        0      0     0
HetFail         NA       0       0        0      0     0
IBSFail         NA      NA       1        0      0     0
isfemale        NA      NA      NA        0      0     0
ismale          NA      NA      NA       NA      0     0
isXXY           NA      NA      NA       NA     NA     0
```

If the procedure did not run, check previous Note.

Indeed, in the updated data set few markers do not pass our QC criteria and we need to drop a few markers. This is done by

```
> data2 <- data2[qc2$idok, qc2$snpok]
```

This is going to be our final analysis data set, therefore let us attach the phenotypic data to the search path, then we do not need to type `data2@phdata$...` to access `dm2` status or other variables:

```
> detach(data1@phdata)
> attach(data2@phdata)
```

5.4 GWA association analysis

Let us start again with descriptives of the phenotypic and marker data

```
> descriptives.trait(data2, by = dm2)
```

```
       No(by.var=1)    Mean     SD No(by.var=0)    Mean     SD   Ptt   Pkw
id               78      NA     NA           48      NA     NA    NA    NA
sex              78   0.603  0.493           48   0.438  0.501 0.074 0.072
age              78  50.508 12.406           48  46.378 13.865 0.095 0.103
dm2              78      NA     NA           48      NA     NA    NA    NA
height           78 170.454 10.580           47 167.988  8.610 0.158 0.212
weight           78  94.047 26.806           47  77.273 17.427 0.000 0.000
diet             78   0.064  0.247           48   0.062  0.245 0.972 0.972
bmi              78  32.188  8.291           47  27.485  6.539 0.001 0.001
       Pexact
id         NA
sex     0.097
age        NA
dm2        NA
height     NA
weight     NA
diet    1.000
bmi        NA
```

You can see that relation to weight is maintained in this smaller, but hopefully cleaner, data set; moreover, relation to age becomes boundary significant.

If you check descriptives of markers (only HWE part shown)

```
> descriptives.marker(data2)[2]
```

```
$`Cumulative distr. of number of SNPs out of HWE, at different alpha`
     X<=1e-04 X<=0.001 X<=0.01 X<=0.05 all X
No          1    2.000  27.000 111.000  3545
Prop        0    0.001   0.008   0.031     1
```

you can see that the problems with HWE are apparently fixed; we may guess that these were caused by the Wahlund's effect.

Run the score test on the cleaned data by

```
> data2.qt <- qtscore(dm2, data2)
```

and check lambda

```
> data2.qt$lambda

$estimate
[1] 1.029865

$iz0
[1] 0.9956654

$iz2
[1] 0.7809572
```

there is still some inflation, which is explained by the fact that we investigate only few short chromosomes with high LD and few causative variants.

Produce the association analysis plot by

```
> plot(data2.qt, df = "Pc1df")
```

(figure 14).

Produce the scan summary by

```
> descriptives.scan(data2.qt, sort = "Pc1df")

          Chromosome Position   N      effB     P1df    Pc1df     effAB
rs1719133          1  4495479 126 -0.273830 0.000295 0.000361 -0.219091
rs4804634          1  2807417 123 -0.217143 0.001017 0.001204 -0.092504
rs8835506          2  6010852 123  0.228338 0.001097 0.001296  0.311111
rs3925525          2  6008501 126  0.221933 0.001425 0.001673  0.295894
rs3224311          2  6009769 126  0.221933 0.001425 0.001673  0.295894
rs2975760          3 10518480 125  0.219461 0.001542 0.001806  0.250000
rs4534929          1  4474374 125 -0.188960 0.001650 0.001929 -0.152980
rs1013473          1  4487262 126  0.189561 0.001979 0.002302  0.271483
rs8258863          1  4735725 125 -0.381808 0.002520 0.002913 -0.381808
rs1048031          1  4485591 124 -0.187133 0.002715 0.003132 -0.134126
              effBB     P2df
rs1719133 -0.729730 0.000780
rs4804634 -0.394737 0.002320
rs8835506  0.311111 0.001742
rs3925525  0.310036 0.002664
rs3224311  0.310036 0.002664
rs2975760  0.388889 0.005889
rs4534929 -0.401042 0.006290
rs1013473  0.385396 0.005528
rs8258863        NA 0.002520
rs1048031 -0.388158 0.008869
```

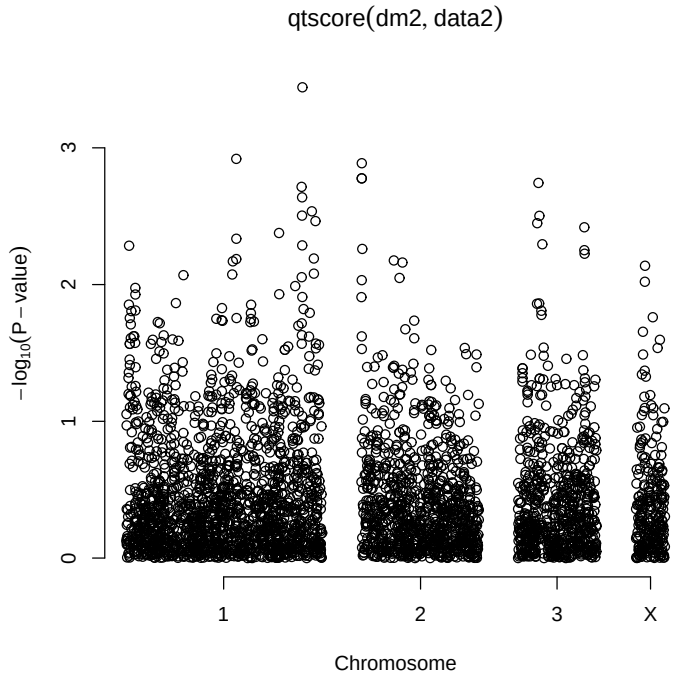

Figure 14: $-log_{10}(CorrectedP - value)$ from the genome scan after the QC procedure

Comparison with the top 10 from the scan before QC shows that results changed substantially with only few markers overlapping.

You can see similar results when accessing empirical GW significance:

```
> data2.qte <- emp.qtscore(dm2, data2)

100%

> descriptives.scan(data2.qte, sort = "Pc1df")

          Chromosome Position   N      effB P1df Pc1df     effAB     effBB P2df
rs1719133          1  4495479 126 -0.273830 0.43  0.50 -0.219091 -0.729730 0.69
rs4804634          1  2807417 123 -0.217143 0.81  0.89 -0.092504 -0.394737 0.96
rs8835506          2  6010852 123  0.228338 0.87  0.89  0.311111  0.311111 0.90
rs3925525          2  6008501 126  0.221933 0.93  0.95  0.295894  0.310036 0.97
rs3224311          2  6009769 126  0.221933 0.93  0.95  0.295894  0.310036 0.97
rs4534929          1  4474374 125 -0.188960 0.95  0.96 -0.152980 -0.401042 1.00
rs2975760          3 10518480 125  0.219461 0.95  0.96  0.250000  0.388889 1.00
rs1013473          1  4487262 126  0.189561 0.97  0.97  0.271483  0.385396 1.00
rs8258863          1  4735725 125 -0.381808 0.97  0.98 -0.381808        NA 1.00
rs5308595          3 10543128 124  0.268559 0.98  0.99  0.279167  0.466667 1.00
```

Again, none of the SNPs hits GW 5% significance. Still, you can see that after QC top markers achieve somewhat "better" significance.

In the last part, we will do several adjusted and stratified analyses. Only empirical P-values will be estimated to make the story shorter. To adjust for sex and age, we can

```
> data2.qtae <- emp.qtscore(dm2 ~ sex + age, data2)
```

```
100%
```

```
> descriptives.scan(data2.qtae)
```

```
          Chromosome Position   N      effB P1df Pc1df     effAB     effBB P2df
rs1719133          1  4495479 126 -0.261763 0.58  0.66 -0.232051 -0.622354 0.93
rs4804634          1  2807417 123 -0.208069 0.98  0.99 -0.088001 -0.378040 1.00
rs3925525          2  6008501 126  0.212271 0.99  1.00  0.296617  0.271922 1.00
rs8835506          2  6010852 123  0.215607 0.99  1.00  0.306910  0.270646 0.97
rs3224311          2  6009769 126  0.212271 0.99  1.00  0.296617  0.271922 1.00
rs7435137          1  4259040 125  0.074654 1.00  1.00  0.108004  0.146743 1.00
rs664063           2  7288020 126 -0.096829 1.00  1.00 -0.055526 -0.616330 1.00
rs546570           2  6120257 125 -0.154982 1.00  1.00  0.133319 -0.021663 1.00
rs7908680          1  2311762 123  0.096321 1.00  1.00  0.096321        NA 1.00
rs166732           1  4716343 125 -0.022318 1.00  1.00 -0.022318        NA 1.00
```

You can see that there is little difference between adjusted and unadjusted analysis, but this is not always the case; adjustment may make your study much more powerful when covariates explain a large proportion of environmental trait variation.

Finally, let us do stratified (by BMI) analysis. We will contracts obese ($BMI \geq 30$) cases to all controls.

```
> data2.qtse <- emp.qtscore(dm2 ~ sex + age, data2, ids = ((bmi >
+     30 & dm2 == 1) | dm2 == 0))
```

```
100%
```

```
> descriptives.scan(data2.qtse, sort = "Pc1df")
```

```
          Chromosome Position  N      effB P1df Pc1df     effAB     effBB P2df
rs1891586          1  2297398 89 -0.252553 0.81  0.89 -0.184492 -0.474411    1
rs794264           1  2534738 88  0.228509 0.91  0.97  0.318452  0.494758    1
rs7435137          1  4259040 89  0.081047 1.00  1.00  0.129850  0.155171    1
rs664063           2  7288020 89 -0.064905 1.00  1.00 -0.015982 -0.514381    1
rs546570           2  6120257 89 -0.123912 1.00  1.00  0.109990 -0.013923    1
rs7908680          1  2311762 87  0.230959 1.00  1.00  0.230959        NA    1
rs166732           1  4716343 88  0.007877 1.00  1.00  0.007877        NA    1
rs4257079          1  3455895 89  0.000073 1.00  1.00  0.000073        NA    1
rs5150804          2  7178160 88  0.043921 1.00  1.00  0.001420  0.120755    1
rs3508821          2  7069507 87 -0.079538 1.00  1.00 -0.006042 -0.451233    1
```

Again, noting interesting at GW significance level. If we would have had found something, naturally, we would not known if we mapped a T2D or obesity gene (or a gene for obesity in presence of T2D, or the one for T2D in presence of obesity).

At this point, you acquired the knowledge necessary for the self-exercise. Please close `R` by `q()` command and proceed to the next section.

5.5 GWA exercise

During the exercise, you will work with a larger data set (approximately 1,000 people and 7,000+ SNPs). You are to do complete three-round QC; perform GWA analysis with `dm2` as the outcome of interest and identify 10 SNPs which you would like to take to the stage 2 (replication) scan. You will do replication analysis using a confirmatory data set. If you did everything right, the SNPs which you identified as significant or replicated will be located in know T2D genes.

Please keep in mind that the data are simulated, and do not take your findings too seriously!

Start `R` by going to "Start -> Programs -> R -> R-2.4.1". Load `GenABEL` library by

```
> library(GenABEL)
```

The two data sets we will use in this exercise are part of the `GenABEL` distribution. The first one ("discovery" set) can be loaded by

```
> data(ge03d2)
```

Please move along the lines detailed in the guided exercise and try to answer following questions:

Exercise 29 *How many cases and controls are presented in the original data set?*

Exercise 30 *How many markers are presented in the original data set?*

Exercise 31 *Is there evidence for inflation of the HWE test staistics?*

Exercise 32 *Perform GWA analysis of the raw data, using assimptotic test and plot the results. Try to think how you can produce $\chi^2 - \chi^2$ plot for the P-values on 1 d.f.. What is the estimate of λ for the 1 d.f. test?*

Exercise 33 *Analyse empirical GW significance. How many SNPs pass genome-wide significance threshold, after correction for the inflation factor? Write down the names of these SNPs for further comparison.*

Perform complete three steps of the genetic data QC.

Exercise 34 *How many male turned apparently female?*

Exercise 35 *How many sporadic X errors do you still observe even when the female male and non-X X-markers are removed? (do not forget to `Xfix()` these!)*

Exercise 36 *How many "twin" DNAs did you discover?*

Exercise 37 *How many genetic outliers did you discover?*

After you have finished QC, answer the questions:

Exercise 38 *How many cases and controls are presented in the data after QC?*

Exercise 39 *How many markers are presented in the data after QC?*

Exercise 40 *Is there evidence for inflation of the HWE test staistics?*

Exercise 41 *Perform GWA analysis of the cleaned data, using assimptotic test and plot the results. What is the estimate of λ for the 1 d.f. test?*

Exercise 42 *Analyse empirical GW significance. How many SNPs pass genome-wide significance threshold, after correction for the inflation factor? Do these SNPs overlap much with the ones ranked at the top before the QC? If not, what could be the reason?*

If time permits, do analysis with adjustment for covariates and stratified analysis.

Select 10 SNPs which you would like to follow-up. Say, you've selected rs1646456, rs7950586, rs4785242, rs4435802, rs2847446, rs946364, rs299251, rs2456488, rs1292700, and rs8183220.

Make a vector of these SNPs with

```
> vec12 <- c("rs1646456", "rs7950586", "rs4785242", "rs4435802",
+     "rs2847446", "rs946364", "rs299251", "rs2456488", "rs1292700",
+     "rs8183220")
```

Load the stage 2 (replicaton) data set by

```
> data(ge03d2c)
```

and select the subset of SNPs you need by

```
> confdat <- ge03d2c[, vec12]
```

Analyse the `confdat` for association with `dm2`.

Exercise 43 *Given the two-stage design, and applying the puristic criteria specified in the lecture, for how many SNPs you can claim a significant finding?*

Exercise 44 *Using the same criteria, for how many SNPs you can claim a replicated finding?*

You can check if any of the SNPs you have identified as significant or replicated are the ones which were simulated to be associated with `dm2` by using the command

```
> show.ncbi(c("snpname1", "snpname2", "snpname3"))
```

where `snpnameX` stands for the name of your identified SNP. The "true" SNPs can be found on NCBI and are located in known T2D genes (just because we used these names to name the "significant" ones).

If time permits, characterise the mode of inheritance of the significant SNPs. You can convert data from `GenABEL` format to the format used by `dgc.genetics` and `genetics` libraries by using `as.genotype()` function. Consult help for details. Please do not attempt to convert more then few dozens SNPs: the format of `genetics` is not compressed, which means conversion may take long and your low-memory computer may even crash if you attempt to convert the whole data set.

If time permits, try to do first round of QC allowing for HWE checks (assume FDR of 0.1 for total sample). In this case, can you still detect stratification in the "cleaned" data?

6 GWA in presence of genetic stratification

Both ethnic admixture and presence of close relationships represents examples of confounding in association analysis. However, the methods to correct for stratification as resulting from mixture of subjects coming from different genetic populations, and methods to correct for family relations may be slightly different, and will be described separately in the next two sections.

6.1 Analysis with ethnic admixture

In previous section we detected genetic stratification by analysis of genomic kinship matrix and excluded genetic outliers from our further analysis. When there are only a few such outliers, exclusion them from analysis is a good option. However, in large studies cases and controls are usually selected across a number of locations and genetic populations, and stratification is expected by design. In such case, analysis of association should account for this stratification.

One of the ways to do that is to perform *structured association* analysis. In such analysis, effect size estimate and its' variance is estimated within each strata separately, and then these estimates are pooled to generate global statistics. The strata can be known from design (e.g. place of birth or ethnicity of parents) or estimated from GWA data.

Let us do structured association analysis using the `data1` data derived in previous section. First, we need to define the variable which will tell what population the study subjects belong to. In previous section, we stored the names of 'outlier' subjects in variable `cl1`:

```
> cl1

[1] "id2097" "id6954" "id2136" "id858"
```

We can use function `%in%` to find out what names of subjects are in `cl1`:

```
> pop <- as.numeric(data1@phdata$id %in% cl1)
> pop

  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [75] 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Now, structured association may be done with `qtscore` function by specifying `strata`:

```
> data1.sa <- qtscore(dm2, data = data1, strata = pop)
```

We can compare results of analysis excluding outliers (black dots) and structured association analysis (green) by

```
> plot(data2.qt, cex = 0.5, pch = 19, ylim = c(1, 5))
> add.plot(data1.sa, col = "green", cex = 1.2)
```

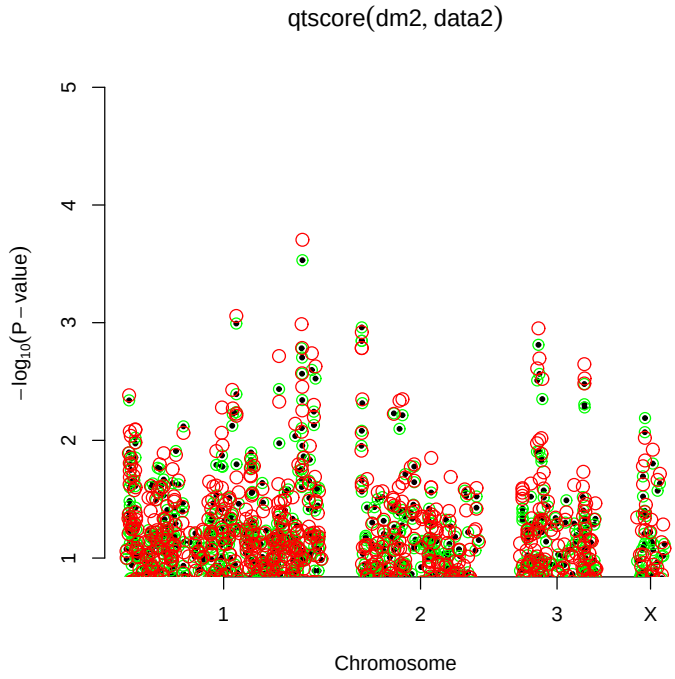

Figure 15: Comparison of structured association analysis (green), method of Price et al. (red) and analysis excluding genetic outliers (solid black).

The resulting plot is presented at figure 15.

In this case, there is very little difference, because all people belonging to the smaller sub-population are cases.

Other way to adjust for genetic (sub)structure is to apply the method of Price et al., which make use of principal components of the genomic kinship matrix to adjust both phenotypes and genotypes for possible stratification. In `GenABEL` , such analysis is done using `egscore` function:

```
> data1.eg <- egscore(dm2, data = data1, kin = data1.gkin)
```

The analysis plot may be added to the previous one by

```
> add.plot(data1.eg, col = "red", cex = 1.3)
```

The resulting plot is presented at figure 15.

Again, the difference between three analysis methods is marginal because there are no highly differentiated SNPs in the data set, and one sub-population is presented by cases only. Still, the signals at chromosome one and three slightly improved, while these at two and X went down.

Exercise 45 *Load and analyse the data set presented in file* `stratified.RData`*. GWA data presented in this file concern a study containing data from several*

popualtions. All these populations originate from the same base population some generations ago. Some of these populations mantained large size and some were small. There was little (2.5%) migration between populations.

Two traits (quat and bint) ara available for analysis. Investigate relations between phenotypes and covariates. Perform association analysis using structured association and `egscore`*. Answer the questions*

1. *How many SNPs and IDs are presented in the data set?*
2. *How many SNPs and IDs pass the quality control?*
3. *Is there evidence for stratification coming from the distribution of GW test for HWE (what is lambda?)*
4. *Is there evidence that the test statistics for trait* `quat` *is inflated (what is lambda?)*
5. *Is there evidence that the test statistics for trait* `bint` *is inflated (what is lambda?)*
6. *How many genetically distinct populations are present in the data set? How many people belong to each population?*
7. *Please make some inferences on the characteristics / history of these populations.*
8. *What is the strongest SNP associated with trait* `quat`*? What model (method and covariates used) gives best results? Is the finding GW-significant?*
9. *What is the strongest SNP associated with trait* `bint`*? What model (method and covariates used) gives best results? Is the finding GW-significant?*

6.2 Analysis of family data

See help for `polygenic`, `mmscore` and `grammar`.

7 Exploring and using public databases

See help for `show.ncbi`

More functions coming soon.

8 Genetic data imputations

With very dense SNP panels available nowadays, linkage disequilibrium between neighbouring SNP is high. This allows us, based on the context, predict genotypes with high degree of accuracy. The genetic data imputations refer to such prediction of genotypes of missing SNPs,

The uses of genetic data imputations are basically two-fold. The first situation concerns imputations of partly missing genotypes of SNPs, typed, in a study, based on the context provided by SNPs typed in the same study. This allows decreasing the proportion of missing SNPs, potentially improving the power of the study. The second situation concerns prediction of genotypes at SNPs, which were not typed in the study, based on the context provided by the SNPs typed and using the data coming from another study with more dense genotyping, which may be potentially done using other population. Though imputations may be done using the same method under the two scenarios, and difference between them is quantitative rather than qualitative, for practical reasons it is useful to distinguish these two situations.

`GenABEL` is well connected with `MACH` imputations software, developed by Y. Li and G. Abecasis. Please read abou `MACH` at http://www.sph.umich.edu/csg/abecasis/MACH/.

8.1 Imputing partly missing genotypes

Assume you have `gwaa.data` object "data" and you want to impute missing genotypes for chromosome 1. For this, you will first need to export chromosome one data in `MACH` format. The `MACH` data format bears remarkable similarity to the format used by `MERLIN`, an excellent linkage analysis program. This is not surprising: `MERLIN` was developed by G. Abecasis. Therefore we can export the data for `MACH` using our function `export.merlin`:

```
> export.merlin(data[,data@gtdata@chromosome=="1"],pedf="chr1.ped",
+                                      dataf="chr1.dat",mapf="chr1.map")
```

where the first argument supplies the data to be exported and others specify the names for the output files containing `MACH` genetic data (pedigree-file), and descriptives (data-file). It will also save map information in file 'chr1.map'; this information is not necessarily used by `MACH`, but is required by `MERLIN`, and also required by `GenABEL` .

Given the data are exported, we can use `MACH` for imputations of missing genotypes. Read `MACH` documentation for details. Basically, under `UNIX`, you need to run command

```
bash> mach1 -p chr1.ped -d chr1.dat --states 200 --rounds 20
                              --mle --mldetails --prefix chr1.out
```

This should produce files, of which we will use `chr1.out.mlgeno`, containing the imputed genotypes and `chr1.out.mlinfo`, containing information about imputations process, including quality score.

Now we need to read the imputed data back into R environment. First, we need convert `MACH` output to `GenABEL` format:

```
> convert.snp.mach(pedf="chr1.out.mlgeno",mapf="chr1.map",
+           info="chr1.out.mlinfo",outf="chr1.raw",quality=0.9)
```

This will convert MACH output to file 'chr1.raw', containing genotypic data in `GenABEL` format. All SNPs with quality < 0.9 will be dropped.

Conversion of data from MACH to GenABEL format may be a lengthy procedure. You can improve the speed much by replacing all "/" to space (" ") in 'chr1.out.mlgeno'.

As soon as conversion is done, you can import the data by

```
> chr1.imp <- load.gwaa.data(phe="pheno.dat",gen="chr1.raw")
```

This assumes that your penotypes are in file "pheno.dat", probably easiest way to generate it is by

```
> save.gwaa.data(data,phen="pheno.dat",geno="tmp.raw")
```

Later, you can even replace your original data with imputed:

```
> data@gtdata@gtps[,data@gtdata@chromosome=="1"] <- chr1.imp@gtdata@gtps
```

8.2 Inferences from other data sets

If you imported your data to GenABEL before version 1.2-6, you will need to re-load your data to GenABEL , to provide the information about actual coding and strand (easily available for Affymetrix). Please check documentation for convert.snp.text, convert.snp.illumina, convert.snp.ped. For the latter two take care that strand="file".

Assume that your data are in gwaa.data object data; you want to impute SNPs in the region between SNP 100 and 500.

Go to HapMap web-site and download phased haplotype data for the region between SNP 100 and 500. We have a script which generates MACH input files from HapMap dump-files (dump2mach.pl, available upon request). Note that in HapMap phased haplotype data, SNP coding is given for "+" strand.

Assume that haplotype and SNP file names are reg1.haplo, reg1.snps (read MACH documentation for details). Check that all of your SNPs (from 100 to 500) are in HapMap data; else generate the vector of indexes/names of the ones which are in HapMap (only these will pass through MACH). Assuming all 500 are in HapMap, export your regional data in MACH format by:

```
> export.merlin(data[,c(100:500)],pedf="reg1.ped",
+              dataf="reg1.dat",mapf="reg1.map",strand="+")
```

To do imputations, in bash type

```
bash> mach1 -p reg1.ped -d reg1.dat -h reg1.haplo -s reg1.snps
           -greedy --rounds 20 --mle --mldetails --prefix reg1.out
```

This should produce files, of which we will use reg1.out.mlgeno, containing the imputed genotypes, and reg1.out.mlinfo, containing information about imputations process, including quality score.

Back in R, convert the data to GenABEL format by:

```
> convert.snp.mach(pedf="reg1.out.mlgeno",mapf="reg1.large.map",
+               info="reg1.out.mlinfo",outf="reg1.raw",quality=0.9)
```

Here, reg1.large.map provides map for all HapMap SNPs in the region (this one is easily made from the dump-file, or with dump2mach.pl) The above command will convert MACH output to `GenABEL` format, dropping all SNPs with quality < 0.9.

Conversion of data from `MACH` to `GenABEL` format may be a lengthy procedure. You can improve the speed much by replacing all "/" to space (" ") in 'chr1.out.mlgeno'.

Now, you can import the data by

```
> reg1.imp <- load.gwaa.data(phe="pheno.dat",gen="reg1.raw")
```

9 Meta-analysis of GWA scans

9.1 Preparing data for meta-analysis

In this section, we will discuss specifics of analysis when meta-analysis is aimed at later stage, and perform simple meta-analysis of two data sets. Let us start with arranging two data sets – basically, we will use cleaned data from the GWA exercise you did in section 5 (Genome-wide association analysis, page 53), and split that is two parts.

```
> data2@gtdata@nids
```

```
[1] 126
```

```
> mdta1 <- data2[1:40, ]
> mdta2 <- data2[41:data2@gtdata@nids, ]
```

We will analyse body mass index. Obviously, in meta-analysis of multiple data sets different individual studies will assess different population, will use slightly different designs, measure different covariates, and so on. Therefore for quantitative traits, for the purpose of future meta-analysis, it becomes conventional to analyse data from individual studies using not the raw data, but pre-adjusted data on the transformed Z-scale (mean of zero and variance of unity). This argument applies only to studies of quantitative traits and to meta-analysis – you may and should report effects on the raw scale (e.g. in centimeters and grams) in analysis of individual studies.

`GenABEL` implements the `ztransform` function for the purposes of Z-transformation. This function takes two (actually three – see help for details) arguments: formula (or just the variable name) and data. `ztransform` function will perform (generalised) linear regression using the specified formula, and will transform the residuals from analysis onto Z-scale by subtracting the mean and division by the standard deviation.

Let us consider what this function does practically. Let us first transform BMI from the first set without using covariates:

```
> x0 <- ztransform(bmi, mdta1)
```

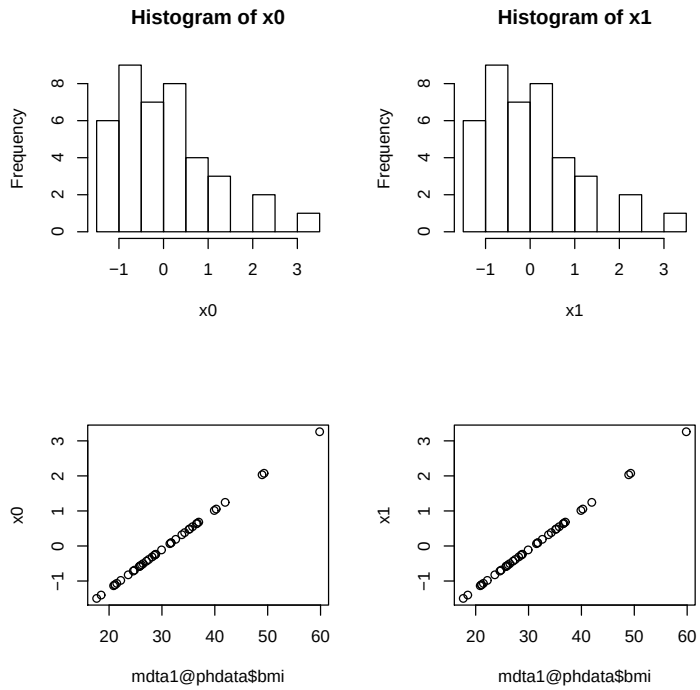

Figure 16: Histogram of Z-transformed BMI and scatter-plots of the raw BMI against Z-transformed BMI. Column 1: Transformation without covariates. Column 2: Transformation with adjustment for age and sex.

The plot of raw against transformed BMI is given at figure 16, column 1. We can also do transformation using sex and age-adjusted residuals with

```
> x1 <- ztransform(bmi ~ sex + age, mdta1)
```

The plot of raw against transformed BMI is given at figure 16, column 2.

Let us analyse Z-transformed, sex and age-adjusted BMI in the two data sets, using `qtscore` function. Analysis of the first study is done with

```
> zbmi1 <- ztransform(bmi ~ sex + age, mdta1)
> qts1 <- qtscore(zbmi1, mdta1)
```

and analysis of the second study is done with

```
> zbmi2 <- ztransform(bmi ~ sex + age, mdta2)
> qts2 <- qtscore(zbmi2, mdta2)
```

The analysis looks very simple – is not it? However, the real difficulty did not begin yet. The point is that different studies may use different strands, and, consequently, allele coding – and one needs to provide this information if this one wants to do right meta-analysis later on!

Let us summarise all the data required for meta-analysis in a new data frame. Let us restrict analysis to 1df test. Than, this data frame should contain (1)

SNP name (2) chromosome (3) position (4) number of people with available data (5) effect of the allele (6) standard error of the effect (7) P-value for the test (8) corrected P-value (we will use Genomic Control) (9) coding, with reference allele coming first (10) strand (11) frequency of the reference allele and, finally, (11) *P-value* from the exact test for HWE (that would be good for QC checks later on).

This can be done by hand, if you prefer greater flexibility, or by running specifically designed `formetascore` function, which does the same automatically.

Let us consider this procedure done by hand. In the list of output parameters, two are not directly available – frequency of the reference allele, which can be computed with

```
> refallfreq <- 1 - summary(mdta1@gtdata)$Q.2
```

and the standard error of the effect estimate. Though `GenABEL` does not report the s.e., it can be computed from the χ^2 value of the test. By definition,

$$\left(\frac{\beta}{SE_\beta}\right)^2 = \chi^2_1$$

and thus given value of the χ^2 test and knowing the effect, we can compute standard error with

$$SE_\beta = \frac{\beta}{\sqrt{\chi^2}}$$

Thus, s.e. is

```
> seeff <- qts1$effB/sqrt(qts1$chi2.1df)
```

At this moment we can arrange the required data frame:

```
> mdf1 <- data.frame(name = qts1$snpnames, chrom = qts1$chromosome,
+     pos = qts1$map, N = qts1$N, beta = qts1$effB, seb = seeff,
+     P = qts1$P1df, Pgc = qts1$Pc1df, coding = as.character(mdta1@gtdata@coding),
+     strand = as.character(mdta1@gtdata@strand), freq = refallfreq)
```

Let us inspect the first 10 raws of the resulting output:

```
> mdf1[1:5, ]
```

```
              name chrom     pos  N        beta       seb         P       Pgc
rs7435137 rs7435137     1 4259040 39 0.009839088 0.2099860 0.9626281 0.9626281
rs664063   rs664063     2 7288020 40 0.060907200 0.4372373 0.8892130 0.8892130
rs546570   rs546570     2 6120257 40 0.085793076 0.3903124 0.8260221 0.8260221
rs7908680 rs7908680     1 2311762 40 0.687189053 0.5927490 0.2463235 0.2463235
rs166732   rs166732     1 4716343 39 0.357996993 0.7258854 0.6218807 0.6218807
          coding strand     freq
rs7435137     CT      - 0.500000
rs664063      GC      - 0.925000
rs546570      TA      + 0.100000
rs7908680     CA      - 0.962500
rs166732      TG      - 0.974359
```

The same result can be achieved by using `formetascore` function, which simply performs the above steps, including transformation, for you:

```
> mdf1 <- formetascore(bmi ~ sex + age, mdta1)
```

You can see that results are exactly the same as previously:

```
> mdf1[1:5, ]
```

```
              name chromosome position strand allele1 allele2 effallele
rs7435137 rs7435137          1  4259040      -       C       T         T
rs664063   rs664063          2  7288020      -       G       C         C
rs546570   rs546570          2  6120257      +       T       A         A
rs7908680 rs7908680          1  2311762      -       C       A         A
rs166732   rs166732          1  4716343      -       T       G         G
          effallelefreq  n        beta    sebeta         p       pgc    pexhwe
rs7435137    0.50000000 39 0.009839088 0.2099860 0.9626281 0.9626281 0.5210851
rs664063     0.07500000 40 0.060907200 0.4372373 0.8892130 0.8892130 1.0000000
rs546570     0.90000000 40 0.085793076 0.3903124 0.8260221 0.8260221 1.0000000
rs7908680    0.03750000 40 0.687189053 0.5927490 0.2463235 0.2463235 1.0000000
rs166732     0.02564103 39 0.357996993 0.7258854 0.6218807 0.6218807 1.0000000
           call
rs7435137 0.975
rs664063  1.000
rs546570  1.000
rs7908680 1.000
rs166732  0.975
```

To write all the data to a file, we can use standard R `write.table` function:

```
> write.table(mdf1, file = "RData/mdf1.dat")
```

Similar analysis is applied to the second data set:

```
> mdf2 <- formetascore(bmi ~ sex + age, mdta2)
```

We can inspect the first 10 lines of the output with

```
> mdf2[1:5, ]
```

```
              name chromosome position strand allele1 allele2 effallele
rs7435137 rs7435137          1  4259040      -       C       T         T
rs664063   rs664063          2  7288020      -       G       C         C
rs546570   rs546570          2  6120257      +       T       A         A
rs7908680 rs7908680          1  2311762      -       C       A         A
rs166732   rs166732          1  4716343      -       T       G         G
          effallelefreq  n        beta    sebeta         p       pgc    pexhwe
rs7435137    0.47647059 85  0.17859624 0.1661182 0.2823232 0.3086500 0.1950153
rs664063     0.11176471 85  0.21738279 0.2428760 0.3707678 0.3966962 1.0000000
rs546570     0.95238095 84 -0.48603282 0.3699863 0.1889638 0.2135234 1.0000000
rs7908680    0.01219512 82 -0.03669903 0.7190409 0.9592946 0.9614531 1.0000000
rs166732     0.02941176 85 -0.50183831 0.4582576 0.2734724 0.2997433 1.0000000
             call
```

```
rs7435137 1.0000000
rs664063  1.0000000
rs546570  0.9882353
rs7908680 0.9647059
rs166732  1.0000000
```

For a binary trait, you may want to get similar output, however, you do not want to transform the data prior to analysis. In this case, use option `transform="no"`, for example:

```
> x <- formetascore(dm2 ~ sex + age + bmi, mdta2, trans = "no")
> x[1:5, ]
```

```
              name chromosome position strand allele1 allele2 effallele
rs7435137 rs7435137          1  4259040      -       C       T         T
rs664063   rs664063          2  7288020      -       G       C         C
rs546570   rs546570          2  6120257      +       T       A         A
rs7908680 rs7908680          1  2311762      -       C       A         A
rs166732   rs166732          1  4716343      -       T       G         G
          effallelefreq  n        beta     sebeta         p       pgc    pexhwe
rs7435137    0.47647059 85  0.11037448 0.07460195 0.1390035 0.1463467 0.1950153
rs664063     0.11176471 85 -0.09228152 0.10907306 0.3975235 0.4061780 1.0000000
rs546570     0.95238095 84  0.09704482 0.16527084 0.5570784 0.5642849 1.0000000
rs7908680    0.01219512 82 -0.07089154 0.31881317 0.8240331 0.8271879 1.0000000
rs166732     0.02941176 85  0.05592620 0.20579864 0.7858127 0.7896223 1.0000000
              call
rs7435137 1.0000000
rs664063  1.0000000
rs546570  0.9882353
rs7908680 0.9647059
rs166732  1.0000000
```

9.2 Performing meta-analysis

10 Analysis of selected region

10.1 Exploring linkage disequilibrium

See help for `r2fast`.

10.2 Haplotype analysis

Use

```
> gtforld <- as.hsgeno(srdta[, 1:5])
```

to convert part of your SNPs to `haplo.stats` format.

You can also use interface function to do sliding widow analysis

```
> h2 <- scan.haplo("qt1~CRSNP", srdta, snps = c(1:5))
```

10.3 Analysis of interactions

See help for `scan.haplo.2D` and `scan.glm.2D`

A Importing data to GenABEL

As described in section 4.1, GenABEL gwaa.data-class consist of phenotypic data frame and an object of snp.data-class, which contains all genetic data. To import data to GenABEL , you need to prepare two files: one containing the phenotypic, and the other containing genotypic data.

The phenotype file relates study subject IDs with values of covariates and outcomes. In the phenotypic data file, the first line gives a description (variable name) of the data contained in a particular column; the names should better be unique, otherwise R will change them.

The first column of the phenotype file **must** contain the subjects' unique ID, named "id". The IDs listed here, and in the genotypic data file, must be the same. It is recommended that the id names are given in quotation marks (see example below), which will save you a possible troubles with e.g. leading zeros.

There also should also be a column named "sex" and giving sex information (0 = female, 1 = male). Other columns in the file should contain phenotypic information.

Missing values should be coded with "NA"; binary traits should have values 0 or 1.

All subjects present in the genotypic files **must** be listed in the phenotypic file as well, because sex information provided by the phenotypic file is an essential part of the genotypic QC procedure.

An example of few first lines of a phenotype file is as follows:

```
id          sex age    bt1 qt   qt1
"cd289982"  0   30.33  NA  NA   3.93
"cd325285"  0   36.514 1   0.49 3.61
"cd357273"  1   37.811 0   1.65 5.30
"cd872422"  1   20.393 0   1.95 4.07
"cd1005389" 1   28.21  1   0.35 3.90
```

This file tells us that, for example, person 325286 is female (0 in second column), and she has "1" (usually this means a "case") value for the trait "bt1", so on. Person 289982 has measurements only for sex, age and qt1, while other measurements are missing (NA, Not Available).

The second file you need should contains genotypic data. As described in section 4.1 (General description of `gwaa.data-class`, page 37), GenABEL snp.data-class contains different types of information. For every SNP, information on map position, chromosome, and strand should be provided. For every person, every SNP genotype should be provided. GenABEL provides a number of function to convert these data from different formats to the internal GenABEL raw format. We will first consider our preferred format, which we informally call "Illumina"-like.

A.1 Converting from preferred format

We will consider use of `convert.snp.illumina` procedure; details of other procedures are given later. Note that what we call "illumina" format is not really a proprietary format from that company, it is just one of the possible text output format from the Illumina BeadStudio; similar formats are accepted/generated by HapMap and Affymetrix.

The file of the "Illumina" format contains SNPs in rows and IDs in columns. The first line is a "header", containing column names. The first three columns should contain information on SNP name, chromosome, and position. There is an optional (though highly recommended!) fourth column, containing strand information (acceptable codes: "+", "-", "u", the last stands for "unknown"). After that column, each of the residual ones corresponds to an individual, with ID as the column name. Genotypes should be presented by two consecutive characters (no separator).

An example of few first lines of the "illumina" genotypic file is as follows:

```
name    chr    pos strand "cd289982" "cd325285" "cd357273" "cd872422" "cd1005389"
rs1001    1   1235      +        AA         AG         AG         AA          GG
rs6679    9   2344      +        GT         GG         GG         TG          GG
rs2401   22   3455      +        AA         CC         CC         CC          AC
rs123     X  32535      -        TT         GT         TT         TT          TT
rs6679   XY   2344      -        GT         GG         GG         TG          GG
rs876     Y  23556      +        00         00         TT         GG          TT
mitoA1   mt  24245      -        AA         CC         00         00          00
```

It is clear that is not quite conventional Illumina file – because in BeadStudio the alleles are reported using the "top" strand; rather, this is an Affymetrix or HapMap-type of a file. Anyways, this file contains all required genotypic information, and this file format is the preferred one for import. Assume that the file with the genotypic data is called "gen0.illu", and is stored in the directory "RData". You can convert the data to `GenABEL` raw format by

```
> convert.snp.illumina(inf = "RData/gen0.illu", out = "RData/gen0i.raw",
+     strand = "file")

Reading genotypes from file 'RData/gen0.illu' ...
Writing to file 'RData/gen0i.raw' ...
... done.
```

Here is the content of the converted file "gen0i.raw" – internal raw data representation:

```
#GenABEL raw data version 0.1
"cd289982" "cd325285" "cd357273" "cd872422" "cd1005389"
rs1001 rs6679 rs2401 rs123 rs6679 rs876 mitoA1
1 9 22 X XY Y mt
1235 2344 3455 32535 2344 23556 24245
04 0c 0f 08 0c 08 0f
01 01 01 02 02 01 02
69 c0
96 40
d5 80
65 40
96 40
07 40
d0 00
```

Note the option `strand="file"` – it is telling `GenABEL` that strand information is provided in the file.

At this moment, you can load the data into `GenABEL` . Assume that the phenotypic file described above is called "phe0.dat" and the converted genotypic file in the raw `GenABEL` format is called "gen0i.raw". You can load the data using the command

```
> df <- load.gwaa.data(phe = "RData/phe0.dat", gen = "RData/gen0i.raw",
+     force = T)

ids loaded...
marker names loaded...
chromosome data loaded...
map data loaded...
allele coding data loaded...
strand data loaded...
genotype data loaded...
snp.data object created...
assignment of gwaa.data object FORCED; X-errors were not checked!
```

The option "force=TRUE" tells that `GenABEL` should load the data even if it fins sex errors.

You can inspect the loaded data; let us first look into phenotypic data by by

```
> df@phdata

                id sex    age bt1   qt  qt1
cd289982   cd289982   0 30.330  NA   NA 3.93
cd325285   cd325285   0 36.514   1 0.49 3.61
cd357273   cd357273   1 37.811   0 1.65 5.30
cd872422   cd872422   1 20.393   0 1.95 4.07
cd1005389 cd1005389   1 28.210   1 0.35 3.90
```

... and than check that the genotypes have imported right:

```
> g0 <- as.character(df@gtdata)
> g0

          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982  "A/A"  "G/T"  "A/A"  "G/T"  "T/T" "A/A"  NA
cd325285  "A/G"  "G/G"  "C/C"  "G/G"  "T/G" "C/C"  NA
cd357273  "A/G"  "G/G"  "C/C"  "G/G"  "T/T" NA     "T/T"
cd872422  "A/A"  "G/T"  "C/C"  "G/T"  "T/T" NA     "G/G"
cd1005389 "G/G"  "G/G"  "C/A"  "G/G"  "T/T" NA     "T/T"

> as.character(df@gtdata@strand)

rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
   "+"    "+"    "+"    "-"    "-"    "-"    "+"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
  "AG"   "GT"   "CA"   "GT"   "TG"   "CA"   "TG"
```

In a real Illumina file, a coding on the TOP strand is supplied. Then, the file will normally look like

```
name    chr    pos "cd289982" "cd325285" "cd357273" "cd872422" "cd1005389"
rs1001    1   1235        AA         AG         AG         AA          GG
rs6679    9   2344        GT         GG         GG         TG          GG
rs2401   22   3455        AA         CC         CC         CC          AC
rs123     X  32535        TT         GT         TT         TT          TT
rs6679   XY   2344        GT         GG         GG         TG          GG
rs876     Y  23556        00         00         TT         GG          TT
mitoA1   mt  24245        AA         CC         00         00          00
```

and the conversion command will be

```
> convert.snp.illumina(inf = "RData/gen0.illuwos", out = "RData/gen0iwos.raw",
+     strand = "+")

Reading genotypes from file 'RData/gen0.illuwos' ...
Writing to file 'RData/gen0iwos.raw' ...
... done.
```

In this particular data set, after conversion, the "+" strand will actually mean not "forward", but TOP – something you should remember for this particular data. The resulting file will look like this:

You can load the data with

```
> df <- load.gwaa.data(phe = "RData/phe0.dat", gen = "RData/gen0iwos.raw",
+     force = T)

ids loaded...
marker names loaded...
chromosome data loaded...
map data loaded...
allele coding data loaded...
strand data loaded...
genotype data loaded...
snp.data object created...
assignment of gwaa.data object FORCED; X-errors were not checked!
```

Obviously, the "strand" is always "+" (here it means TOP):

```
> g1 <- as.character(df@gtdata)
> g1

          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982  "A/A"  "G/T"  "A/A"  "G/T"  "T/T" "A/A"  NA
cd325285  "A/G"  "G/G"  "C/C"  "G/G"  "T/G" "C/C"  NA
cd357273  "A/G"  "G/G"  "C/C"  "G/G"  "T/T" NA     "T/T"
cd872422  "A/A"  "G/T"  "C/C"  "G/T"  "T/T" NA     "G/G"
cd1005389 "G/G"  "G/G"  "C/A"  "G/G"  "T/T" NA     "T/T"

> as.character(df@gtdata@strand)
```

```
rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
   "+"    "+"    "+"    "+"    "+"    "+"    "+"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
  "AG"   "GT"   "CA"   "GT"   "TG"   "CA"   "TG"
```

We can see that the genotypes are identical to ones we imported previously, as should be the case:

```
> g0 == g1

          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982    TRUE   TRUE   TRUE   TRUE  TRUE   TRUE    NA
cd325285    TRUE   TRUE   TRUE   TRUE  TRUE   TRUE    NA
cd357273    TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
cd872422    TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
cd1005389   TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
```

A.2 Converting PLINK tped files

PLINK TPED (transposed-pedigree) format bears notable similarity to what we call "Illumina" format, with few exceptions: (1) there is no header line giving field names (and therefore IDs are stored in a separate file) (2) the first column gives chromosome, second – SNP name, third *genetic* map (usually kept as zeroes), the fourth – physical position, and, starting with the fifth column, genotypic data are listed, (3) finally, within a genotypes, alleles are separated with a space. In TPED format, the data we already worked with would look like

```
 1   rs1001   0    1235   A A   A G   A G   A A   G G
 9   rs6679   0    2344   G T   G G   G G   T G   G G
22   rs2401   0    3455   A A   C C   C C   C C   A C
 X    rs123   0   32535   T T   G T   T T   T T   T T
XY   rs6679   0    2344   G T   G G   G G   T G   G G
 Y    rs876   0   23556   0 0   0 0   T T   G G   T T
mt   mitoA1   0   24245   A A   C C   0 0   0 0   0 0
```

Obviously, a separate file is needed to keep correspondence between genotypes and IDs. This file emulated standard pedigree file without a header line. The file, conventionally called a TFAM-file, should contain six columns, corresponding to pedigree ID, ID, father, mother, sex, and affection. Only the second column is used by `GenABEL` – please make sure you use unique IDs. Consequently, it does not matter what pedigree ID, father/mother, sex, or affection status you assign in the file – the real information is coming from the phenotypic data file. The TFAM file for our data will look like this:

```
1  cd289982 0 0 1 0
1  cd325285 0 0 1 0
1  cd357273 0 0 1 0
1  cd872422 0 0 1 0
1 cd1005389 0 0 1 0
```

You can convert the data from PLINK TPED format to the GenABEL format using command convert.snp.tped:

```
> convert.snp.tped(tped = "RData/gen0.tped", tfam = "RData/gen0.tfam",
+     out = "RData/gen0tped.raw", strand = "+")

Reading individual ids from file 'RData/gen0.tfam' ...
... done.  Read 5 individual ids from file 'RData/gen0.tfam'
Reading genotypes from file 'RData/gen0.tped' ...
...done.  Read 7 SNPs from file 'RData/gen0.tped'
Writing to file 'RData/gen0tped.raw' ...
... done.
```

and load the data with

```
> df <- load.gwaa.data(phe = "RData/phe0.dat", gen = "RData/gen0tped.raw",
+     force = T)

ids loaded...
marker names loaded...
chromosome data loaded...
map data loaded...
allele coding data loaded...
strand data loaded...
genotype data loaded...
snp.data object created...
assignment of gwaa.data object FORCED; X-errors were not checked!
```

Obviously, the "strand" is always "+" (meaning TOP):

```
> g1 <- as.character(df@gtdata)
> g1

          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982  "A/A"  "G/T"  "A/A"  "G/T"  "T/T" "A/A"  NA
cd325285  "A/G"  "G/G"  "C/C"  "G/G"  "T/G" "C/C"  NA
cd357273  "A/G"  "G/G"  "C/C"  "G/G"  "T/T" NA     "T/T"
cd872422  "A/A"  "G/T"  "C/C"  "G/T"  "T/T" NA     "G/G"
cd1005389 "G/G"  "G/G"  "C/A"  "G/G"  "T/T" NA     "T/T"

> as.character(df@gtdata@strand)

rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
   "+"    "+"    "+"    "+"    "+"    "+"    "+"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
  "AG"   "GT"   "CA"   "GT"   "TG"   "CA"   "TG"
```

We can see that the genotypes are identical to ones we imported previously, as should be the case:

```
> g0 == g1
```

```
          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982    TRUE   TRUE   TRUE   TRUE  TRUE   TRUE    NA
cd325285    TRUE   TRUE   TRUE   TRUE  TRUE   TRUE    NA
cd357273    TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
cd872422    TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
cd1005389   TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
```

A.3 Converting linkage-like files

Linkage-like files, also known as pre-makeped files, or pedigree files, represent a historic format which dates back to the time when only few markers could be typed – thus the number of subjects was usually greater than the number of markers. In that situation, it was natural and obvious to keep IDs in rows and markers in columns. In the first six columns, standard linkage-like file would contain pedigree ID, ID, father's ID, mother's ID, sex (coded as 1 = male and 2 = female), and affection status (0 = unknown, 1 = unaffected, 2 = affected). In the following columns, genotypic information is provided. Alleles of the same genotype could be separated by a space, or by a slash ("/"). Thus the data we are working with could be presented as

```
1  cd289982 0 0 1 0   A A   G T   A A   T T   G T   0 0   A A
1  cd325285 0 0 1 0   A G   G G   C C   G T   G G   0 0   C C
1  cd357273 0 0 1 0   A G   G G   C C   T T   G G   T T   0 0
1  cd872422 0 0 1 0   A A   T G   C C   T T   T G   G G   0 0
1 cd1005389 0 0 1 0   G G   G G   A C   T T   G G   T T   0 0
```

As you can see, this file misses header line, and information what are the SNP names, position, etc. should be provided in a separate MAP-file. `GenABEL` accepts map in Merlin format, and an extended format. A map in Merlin format consist of header line, giving column names, and three columns with chromosome, name and position information, for example:

```
chr  name    pos
1    rs1001  1235
9    rs6679  2344
22   rs2401  3455
X    rs123   32535
XY   rs6679  2344
Y    rs876   23556
mt   mitoA1  24245
```

The data can be converted to the internal `GenABEL` format with

```
> convert.snp.ped(ped = "RData/gen0.ped", map = "RData/map0.dat",
+     out = "RData/gen0pedwos.raw", strand = "+")

Reading map from file 'RData/map0.dat' ...
... done.  Read positions of 7 markers from file 'RData/map0.dat'
Reading genotypes from file 'RData/gen0.ped' ...
...done.  Read information for 5 people from file 'RData/gen0.ped'
Analysing marker information ...
Writing to file 'RData/gen0pedwos.raw' ...
... done.
```

and loaded with

```
> df <- load.gwaa.data(phe = "RData/phe0.dat", gen = "RData/gen0pedwos.raw",
+     force = T)

ids loaded...
marker names loaded...
chromosome data loaded...
map data loaded...
allele coding data loaded...
strand data loaded...
genotype data loaded...
snp.data object created...
assignment of gwaa.data object FORCED; X-errors were not checked!
```

We can inspect the genotypic data and check that conversion results are identical to previous runs with

```
> g1 <- as.character(df@gtdata)
> g1

          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982  "A/A"  "G/T"  "A/A"  "G/T"  "T/T" "A/A"  NA
cd325285  "A/G"  "G/G"  "C/C"  "G/G"  "T/G" "C/C"  NA
cd357273  "A/G"  "G/G"  "C/C"  "G/G"  "T/T" NA     "T/T"
cd872422  "A/A"  "G/T"  "C/C"  "G/T"  "T/T" NA     "G/G"
cd1005389 "G/G"  "G/G"  "C/A"  "G/G"  "T/T" NA     "T/T"

> as.character(df@gtdata@strand)

rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
   "+"    "+"    "+"    "+"    "+"    "+"    "+"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
  "AG"   "GT"   "CA"   "GT"   "TG"   "CA"   "TG"

> g0 == g1

          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982    TRUE   TRUE   TRUE   TRUE  TRUE   TRUE    NA
cd325285    TRUE   TRUE   TRUE   TRUE  TRUE   TRUE    NA
cd357273    TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
cd872422    TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
cd1005389   TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
```

If you are willing to import strand information, you can make use of the *extended* map format. In this format the strand information is added to the map-file:

```
chr  name    pos    strand coding
1    rs1001  1235   +      AG
9    rs6679  2344   +      TG
22   rs2401  3455   +      AC
X    rs123   32535  -      GT
XY   rs6679  2344   -      GT
Y    rs876   23556  +      GT
mt   mitoA1  24245  -      AC
```

The data can be converted to the internal `GenABEL` format with

```
> convert.snp.ped(ped = "RData/gen0.ped", map = "RData/emap0.dat",
+     out = "RData/gen0ped.raw", strand = "file")

Reading map from file 'RData/emap0.dat' ...
... done.  Read positions of 7 markers from file 'RData/emap0.dat'
Reading genotypes from file 'RData/gen0.ped' ...
...done.  Read information for 5 people from file 'RData/gen0.ped'
Analysing marker information ...
Writing to file 'RData/gen0ped.raw' ...
... done.
```

Note that option `strand==file` was used to specify that the extended map format should be used. The data can be loaded with

```
> df <- load.gwaa.data(phe = "RData/phe0.dat", gen = "RData/gen0ped.raw",
+     force = T)

ids loaded...
marker names loaded...
chromosome data loaded...
map data loaded...
allele coding data loaded...
strand data loaded...
genotype data loaded...
snp.data object created...
assignment of gwaa.data object FORCED; X-errors were not checked!
```

We can inspect the genotypic data and check that conversion results are identical to previous runs with

```
> g1 <- as.character(df@gtdata)
> g1

          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982  "A/A"  "G/T"  "A/A"  "G/T"  "T/T" "A/A"  NA
cd325285  "A/G"  "G/G"  "C/C"  "G/G"  "T/G" "C/C"  NA
cd357273  "A/G"  "G/G"  "C/C"  "G/G"  "T/T" NA     "T/T"
cd872422  "A/A"  "G/T"  "C/C"  "G/T"  "T/T" NA     "G/G"
cd1005389 "G/G"  "G/G"  "C/A"  "G/G"  "T/T" NA     "T/T"

> as.character(df@gtdata@strand)
```

```
rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
   "+"    "+"    "+"    "-"    "-"    "-"    "+"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401 rs6679  rs123 mitoA1  rs876
  "AG"   "GT"   "CA"   "GT"   "TG"   "CA"   "TG"

> g0 == g1

          rs1001 rs6679 rs2401 rs6679 rs123 mitoA1 rs876
cd289982    TRUE   TRUE   TRUE   TRUE  TRUE   TRUE    NA
cd325285    TRUE   TRUE   TRUE   TRUE  TRUE   TRUE    NA
cd357273    TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
cd872422    TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
cd1005389   TRUE   TRUE   TRUE   TRUE  TRUE     NA  TRUE
```

A.4 Converting from MACH format

The data from MACH format can be converted by using `convert.snp.mach`. This function actually calls `convert.snp.ped` in specific format. MACH software is widely used for SNP imputations. For our needs we consider two files produced by MACH: pedigree file with (the imputed) genotypic data, and info-file, containing information about quality of imputations for particular SNP.

SEE HELP FOR `convert.snp.mach` for further details.

A.5 Converting from text format

B Answers to exercises

B.1 Exercise 3:

For the first person id is "p1" and sex code is 1 (1=male, 2=female)

```
> srdta@gtdata@idnames[1]

[1] "p1"

> srdta@gtdata@male[1]

p1
 1
```

For the 22nd person id is "p22" and sex code is 1:

```
> srdta@gtdata@idnames[22]

[1] "p22"

> srdta@gtdata@male[22]

p22
  1
```

Among first 100 subjects, there are 53 males:

```
> sum(srdta@gtdata@male[1:100])

[1] 53
```

Among 4th hundred subjects there are 45 females:

```
> 100 - sum(srdta@gtdata@male[301:400])

[1] 45
```

Male proportion among first 1000 people is

```
> mean(srdta@gtdata@male[1:1000])

[1] 0.508
```

Female proportion among second 1000 people is

```
> 1 - mean(srdta@gtdata@male[1001:2000])

[1] 0.476
```

Name, chromosome and map position of the 33rd marker are:

```
> srdta@gtdata@snpnames[33]

[1] "rs422"

> srdta@gtdata@chromosome[33]
```

```
rs422
    1
Levels: 1
```

```
> srdta@gtdata@map[33]
```

```
 rs422
105500
```

The map positions for and distance between markers 25 and 26 are:

```
> pos25 <- srdta@gtdata@map[25]
> pos25
```

```
rs365
91250
```

```
> pos26 <- srdta@gtdata@map[26]
> pos26
```

```
rs372
92750
```

```
> pos26 - pos25
```

```
rs372
 1500
```

B.2 Exercise 6:

Value of the 4th variable of person 75:

```
> srdta@phdata[75, 4]
```

```
[1] -0.04
```

Value for the variable 1 is

```
> srdta@phdata[75, 1]
```

```
[1] "p75"
```

Also, if we check first 10 elements we see

```
> srdta@phdata[1:10, 1]
```

```
 [1] "p1"  "p2"  "p3"  "p4"  "p5"  "p6"  "p7"  "p8"  "p9"  "p10"
```

This is personal ID.

The sum for variable 2 is

```
> sum(srdta@phdata[, 2])
```

```
[1] 1275
```

This is sex variable.

B.3 Exercise 7:

To obtain the number of people with age >65 y.o., you can use any of the following

```
> sum(srdta@phdata$age > 65)

[1] 48

> vec <- which(srdta@phdata$age > 65)
> length(vec)

[1] 48
```

To get sex of these people use any of:

```
> sx65 <- srdta@phdata$sex[srdta@phdata$age > 65]
> sx65

 [1] 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0
[39] 1 0 1 0 0 0 0 1 1 1

> sx65 <- srdta@phdata$sex[vec]
> sx65

 [1] 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0
[39] 1 0 1 0 0 0 0 1 1 1
```

Thus, number of males is:

```
> sum(sx65)

[1] 26
```

To conclude, the proportion of male is 0.541666666666667

Distribution of `qt3` in people younger and older than 65 are:

```
> summary(srdta@phdata$qt3[vec])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.730   2.690   3.480   3.499   4.265   5.840

> sd(srdta@phdata$qt3[vec], na.rm = TRUE)

[1] 1.128701

> young <- which(srdta@phdata$age < 65)
> summary(srdta@phdata$qt3[young])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  -1.97    1.83    2.58    2.59    3.35    6.34   11.00

> sd(srdta@phdata$qt3[young], na.rm = TRUE)

[1] 1.093374
```

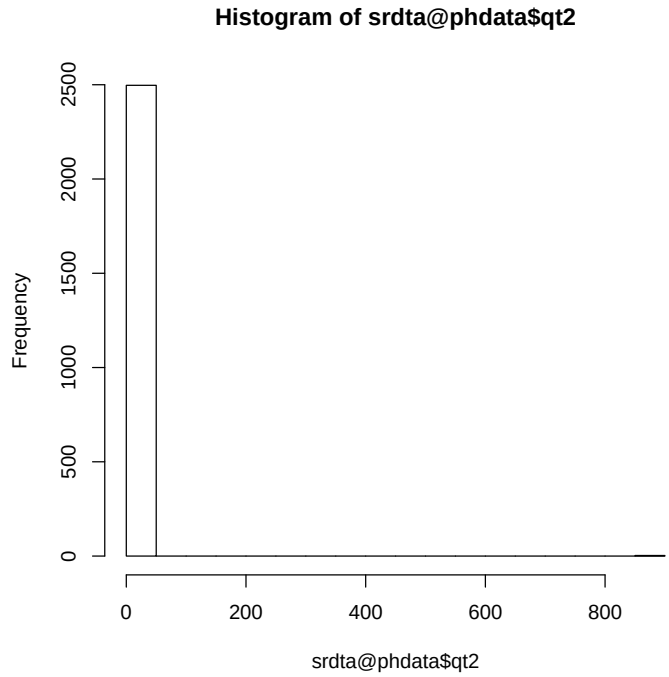

Figure 17: Histogram of qt2

B.4 Exercise 9:

```
> summary(srdta@phdata$age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  24.10   45.10   50.00   50.04   54.80   71.60
```

The histogram for qt2 looks strange: it seems there are few very strong outliers (figure 17) You can also see that with `summary`:

```
> summary(srdta@phdata$qt2)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   4.220   5.045   6.122   5.910 888.000
```

B.5 Exercise 10:

How many SNPs are described in this data frame?

```
> attach(popdat)
```

```
        The following object(s) are masked from data2@phdata :

         sex
```

```
> names(popdat)
```

```
 [1] "subj"  "sex"   "aff"   "qt"    "snp1"  "snp2"  "snp3"  "snp4"  "snp5"
[10] "snp6"  "snp7"  "snp8"  "snp9"  "snp10"
```

The answer is 10 snps

B.6 Exercise 11:

What is the frequency (proportion) of snp1 allele A? What is its frequency in these affected (aff==1)?

```
> summary(snp1)

Number of samples typed: 2374 (95%)

Allele Frequency: (2 alleles)
   Count Proportion
A   3462       0.73
B   1286       0.27
NA   252         NA


Genotype Frequency:
    Count Proportion
B/B   199       0.08
A/B   888       0.37
A/A  1287       0.54
NA    126         NA

Heterozygosity (Hu)  = 0.3950646
Poly. Inf. Content   = 0.3169762
```

The frequency of A in all subjects is 0.73.

```
> summary(snp1[aff == 1])

Number of samples typed: 519 (94.5%)

Allele Frequency: (2 alleles)
   Count Proportion
A    729        0.7
B    309        0.3
NA    60         NA


Genotype Frequency:
    Count Proportion
B/B    48       0.09
A/B   213       0.41
A/A   258       0.50
NA     30         NA

Heterozygosity (Hu)  = 0.4185428
Poly. Inf. Content   = 0.3307192
```

The frequency of A in affected subjects is 0.7.

B.7 Exercise 12:

How many cases and controls are present?

```
> table(aff)

aff
   0    1
1951  549
```

There are 549 cases and 1951 controls.

B.8 Exercise 13:

If all subjects are used to test HWE, are there any SNPs out of HWE at nominal $P \leq 0.05$? Which ones?

```
> HWE.exact(snp1)

        Exact Test for Hardy-Weinberg Equilibrium

data:  snp1
N11 = 1287, N12 = 888, N22 = 199, N1 = 3462, N2 = 1286, p-value =
0.01083
```

...

```
> HWE.exact(snp10)

        Exact Test for Hardy-Weinberg Equilibrium

data:  snp10
N11 = 1792, N12 = 552, N22 = 40, N1 = 4136, N2 = 632, p-value = 0.7897
```

Only SNP 1 is out of HWE in the total sample.

B.9 Exercise 14:

If only controls are used to test the SNPs which are out of HWE in total sample, are these still out of HWE?

```
> HWE.exact(snp1[aff == 0])

        Exact Test for Hardy-Weinberg Equilibrium

data:  snp1[aff == 0]
N11 = 1029, N12 = 675, N22 = 151, N1 = 2733, N2 = 977, p-value =
0.008393
```

Yes, SNP 1 is out of HWE also in controls.

B.10 Exercise 15:

Which SNP pairs are in strong LD ($r^2 \geq 0.8$)?

```
> LD(popdat[, 5:14])$"R^2"
```

```
      snp1  snp2  snp3  snp4  snp5  snp6  snp7  snp8  snp9 snp10
snp1    NA 0.016 0.235 0.206 0.258 0.227 0.152 0.117 0.090 0.000
snp2    NA    NA 0.004 0.004 0.005 0.004 0.000 0.000 0.000 0.000
snp3    NA    NA    NA 0.602 0.457 0.346 0.641 0.031 0.042 0.001
snp4    NA    NA    NA    NA 0.803 0.650 0.729 0.027 0.037 0.002
snp5    NA    NA    NA    NA    NA 0.874 0.586 0.034 0.046 0.002
snp6    NA    NA    NA    NA    NA    NA 0.670 0.030 0.040 0.002
snp7    NA    NA    NA    NA    NA    NA    NA 0.020 0.027 0.003
snp8    NA    NA    NA    NA    NA    NA    NA    NA 0.002 0.000
snp9    NA    NA    NA    NA    NA    NA    NA    NA    NA 0.001
snp10   NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
```

SNP pairs 4-5 and 5-6 have $r^2 \geq 0.8$.

B.11 Exercise 16:

For SNPs in strong LD, what is r^2 for separate samples of cases and controls?
For controls,

```
> LD(data.frame(snp4, snp5, snp6)[aff == 0, ])$"R^2"
```

```
     snp4     snp5      snp6
snp4   NA 0.806591 0.6419715
snp5   NA       NA 0.8661005
snp6   NA       NA        NA
```

For cases,

```
> LD(data.frame(snp4, snp5, snp6)[aff == 1, ])$"R^2"
```

```
     snp4      snp5      snp6
snp4   NA 0.7951475 0.6773275
snp5   NA        NA 0.9083237
snp6   NA        NA        NA
```

Note that the fact that LD is higher in cases may mean nothing because the estimates of LD are biased upwards with smaller sample sizes. For example in a small sample (5 people) of controls we expect even higher LD because of strong upward bias:

```
> LD(popdat[which(aff == 0)[1:5], 8:10])$"R^2"
```

```
     snp4      snp5      snp6
snp4   NA 0.9995876 0.9995876
snp5   NA        NA 0.9995876
snp6   NA        NA        NA
```

More elaborate methods, such as that by Zaykin, are required to contrast LD between sample of unequal size.

B.12 Exercise 17:

Is there significant association between affection status and sex? What is P-value for association?

```
> glm(aff ~ sex, family = binomial())

Call:  glm(formula = aff ~ sex, family = binomial())

Coefficients:
(Intercept)          sex
    -1.3197       0.1006

Degrees of Freedom: 2499 Total (i.e. Null);  2498 Residual
Null Deviance:            2632
Residual Deviance: 2631         AIC: 2635
```

There is significant ($P = 0.03$) association.

B.13 Exercise 18:

Is association between the disease and `qt` significant?

```
> glm(aff ~ qt, family = binomial())

Call:  glm(formula = aff ~ qt, family = binomial())

Coefficients:
(Intercept)           qt
   -1.26769     -0.02514

Degrees of Freedom: 2499 Total (i.e. Null);  2498 Residual
Null Deviance:            2632
Residual Deviance: 2632         AIC: 2636
```

There is no significant ($P = 0.6$) association.

B.14 Exercise 19:

Which SNPs are associated with the quantitative trait `qt` at nominal $P \leq 0.05$? Use 2 d.f. test.

```
> summary(lm(qt ~ snp1))

Call:
lm(formula = qt ~ snp1)

Residuals:
     Min       1Q   Median       3Q      Max
-3.52609 -0.66427 -0.01110  0.67648  3.54622

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -0.02846    0.02758  -1.032   0.3022
snp1A/B      0.08200    0.04316   1.900   0.0575 .
snp1B/B      0.18644    0.07536   2.474   0.0134 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9893 on 2371 degrees of freedom
  (126 observations deleted due to missingness)
Multiple R-squared: 0.00335,        Adjusted R-squared: 0.002509
F-statistic: 3.985 on 2 and 2371 DF,  p-value: 0.01873
```

...

```
> summary(lm(qt ~ snp10))

Call:
lm(formula = qt ~ snp10)

Residuals:
      Min        1Q    Median        3Q       Max
-3.586464 -0.677484  0.001935  0.673270  3.412527

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01915    0.02344   0.817    0.414
snp10A/B     0.01277    0.04829   0.264    0.792
snp10B/B     0.17178    0.15860   1.083    0.279

Residual standard error: 0.9921 on 2381 degrees of freedom
  (116 observations deleted due to missingness)
Multiple R-squared: 0.0005072,        Adjusted R-squared: -0.0003324
F-statistic: 0.6041 on 2 and 2381 DF,  p-value: 0.5467
```

SNPs 1, 4, 5 an 9 are significantly associated at nominal $P \leq 0.05$.

B.15 Exercise 20:

Test each SNP for association with the affection status, using 2 d.f. test. Which SNPs are significantly associated at nominal $P \leq 0.05$? How can you describe the model of action of the significant SNPs?

```
> x <- glm(aff ~ snp5, family = binomial())
> x

Call:  glm(formula = aff ~ snp5, family = binomial())

Coefficients:
(Intercept)      snp5A/A      snp5B/B
    -1.4868       0.2112       0.3387

Degrees of Freedom: 2382 Total (i.e. Null);  2380 Residual
  (117 observations deleted due to missingness)
```

```
Null Deviance:            2440
Residual Deviance: 2431         AIC: 2437
```

```
> anova(x, test = "Chisq")

Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


       Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    2382    2440.40
snp5    2     9.24      2380    2431.16      0.01
```

...

```
> x <- glm(aff ~ snp10, family = binomial())
> x

Call:  glm(formula = aff ~ snp10, family = binomial())

Coefficients:
(Intercept)     snp10A/B     snp10B/B
    -1.3703       0.2909      -0.1803

Degrees of Freedom: 2383 Total (i.e. Null);  2381 Residual
  (116 observations deleted due to missingness)
Null Deviance:            2475
Residual Deviance: 2468         AIC: 2474
```

```
> anova(x, test = "Chisq")

Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    2383    2475.13
snp10    2     6.73      2381    2468.39      0.03
```

The SNPs 5 an 10 are significantly associated at $P \leq 0.05$. The model of action of SNP5 can be ddescribed as recessive (while the risk for AA and AB is not significantly different, there is 1.4 times increased risk for these homozygous for

BB). The SNP 10 demonstrates somewhat weird action with the risk increased in heterozygous AB individuals. However, the confidence interval for BB is large and therefore we can not claim that BB is not increasing the risk.

B.16 Exercise 21:

For the SNPs selected in previous question, test association using additive model. Which SNPs are still associated?

```
> glm(aff ~ as.numeric(snp5), family = binomial())

Call:  glm(formula = aff ~ as.numeric(snp5), family = binomial())

Coefficients:
     (Intercept)  as.numeric(snp5)
         -1.5695            0.1094

Degrees of Freedom: 2382 Total (i.e. Null);  2381 Residual
  (117 observations deleted due to missingness)
Null Deviance:            2440
Residual Deviance: 2438         AIC: 2442

> glm(aff ~ as.numeric(snp10), family = binomial())

Call:  glm(formula = aff ~ as.numeric(snp10), family = binomial())

Coefficients:
      (Intercept)  as.numeric(snp10)
          -1.5539             0.1976

Degrees of Freedom: 2383 Total (i.e. Null);  2382 Residual
  (116 observations deleted due to missingness)
Null Deviance:            2475
Residual Deviance: 2471         AIC: 2475
```

Only SNP 10 is significantly associated under the additive model.

B.17 Exercise 22:

If you adjust the analysis under additive model (question 21) for significant covariates which you discovered in questions 17 and 18, are these findings still significant?

```
> glm(aff ~ sex + snp10, family = binomial())

Call:  glm(formula = aff ~ sex + snp10, family = binomial())

Coefficients:
(Intercept)          sex     snp10A/B     snp10B/B
   -1.41894      0.09513      0.29230     -0.18471

Degrees of Freedom: 2383 Total (i.e. Null);  2380 Residual
```

```
  (116 observations deleted due to missingness)
Null Deviance:            2475
Residual Deviance: 2467         AIC: 2475
```

Yes, SNP 10 becomes even a bit more significantly associated after adjusting for sex.

B.18 Exercise 23:

Test association between `aff` and snp5 and snp10, allowing for the SNPs interaction effect. Use arbitrary (not an additive) model. Do you observe significant interaction? How can you describe the model of concert action of snp5 and snp10?

```
> glm(aff ~ snp5 * snp10, family = binomial())

Call:  glm(formula = aff ~ snp5 * snp10, family = binomial())

Coefficients:
     (Intercept)           snp5A/A           snp5B/B          snp10A/B
        -1.50840          -0.41802           0.33441          -0.01403
        snp10B/B  snp5A/A:snp10A/B  snp5B/B:snp10A/B  snp5A/A:snp10B/B
        -0.14983           1.48369           0.12989           0.82348
snp5B/B:snp10B/B
        -0.28562

Degrees of Freedom: 2268 Total (i.e. Null);  2260 Residual
  (231 observations deleted due to missingness)
Null Deviance:            2282
Residual Deviance: 2243         AIC: 2261
```

It appears that SNP10 genotype is only relevant in these who are homozygous for the low-risk A allele at the SNP5; in such cases SNP 10 allele B is risk increasing. In these homozygous for SNP 5 A, we observe highly significant increase in risk for heterozygotes for SNP10 and increased (though not significantly) risk for SNP 10 BB.

100

B.19 Exercise 24:

```
> data(srdta)
```

Number of people:

```
> srdta@gtdata@nids

[1] 2500
```

Number of males:

```
> sum(srdta@gtdata@male)

[1] 1275
```

Number of females:

```
> srdta@gtdata@nids - sum(srdta@gtdata@male)
```

```
[1] 1225
```

... or you could get the same answer like this[4]:

```
> sum(srdta@gtdata@male == 0)
```

```
[1] 1225
```

The proportion of males can be computed using above results

```
> sum(srdta@gtdata@male)/srdta@gtdata@nids
```

```
[1] 0.51
```

or by using `mean()` function:

```
> mean(srdta@gtdata@male)
```

```
[1] 0.51
```

B.20 Exercise 25:

The names of markers located after 2,490,000 b.p. are

```
> vec <- (srdta@gtdata@map > 2490000)
> srdta@gtdata@snpnames[vec]
```

```
[1] "rs9273" "rs9277" "rs9279" "rs9283"
```

Between 1,100,000 and 1,105,000 b.p.:

```
> vec <- (srdta@gtdata@map > 1100000 & srdta@gtdata@map < 1105000)
> srdta@gtdata@snpnames[vec]
```

```
[1] "rs4180" "rs4186" "rs4187"
```

B.21 Exercise 26:

To learn what allele of "rs114" is the reference you need to run

```
> as.character(srdta@gtdata@coding["rs114"])
```

```
<NA>
"AT"
```

Here, the first ("A") allele is the reference and thus the second ("T") is the effective one. Remember that when using `as.numeric` function to convert the genotypes to human-readable and R-operatable format, the homozygotes for reference will be coded as "0", heterozygotes as "1" and the non-reference ("effective") homozygotes will be coded as "2":

[4]This is something covered later in the section **??** ("**??**")

```
> table(as.character(srdta@gtdata[, "rs114"]), as.numeric(srdta@gtdata[,
+     "rs114"]))

        0    1    2
  A/A 1868    0    0
  A/T    0  491    0
  T/T    0    0   34
```

To compute frequency of the effective allele of SNP "rs114" in total sample, you can go two ways. First, we can try to take a sum of all rs114 genotypes and divide it by twice the number of people:

```
> a <- as.numeric(srdta@gtdata[, "rs114"])
> sum(a)

[1] NA
```

This, however, returns NA, because some of the genotypes are missing. We can deal with this problem by running `sum()` with the option `na.rm=TRUE`:

```
> sum(a, na.rm = T)

[1] 559
```

However, now we do not know what was the number of people for whom the genotype was measured!

An easier way would be to compute mean value of rs114 with the `mean( ... ,na.rm=TRUE)` function and divide it by 2:

```
> mean(a, na.rm = T)/2

[1] 0.116799
```

To compute frequency of the effective allele of "rs114" in males, you can use

```
> amale <- as.numeric(srdta@gtdata[srdta@phdata$sex == 1, "rs114"])
> mean(amale, na.rm = T)/2

[1] 0.1164216
```

To compute frequency of the effective allele in females, you can use

```
> afemale <- as.numeric(srdta@gtdata[srdta@phdata$sex == 0, "rs114"])
> mean(afemale, na.rm = T)/2

[1] 0.1171942
```

Actually, the problem that we do not know how many people are measured, can be easily dealt with. This can be done by using `is.na(A)` function which returns true when some element of `A` is not measured. Thus, the number of people with measured genotype for "rs114" is

```
> a <- as.numeric(srdta@gtdata[, "rs114"])
> sum(!is.na(a))
```

```
[1] 2393
```

And the allele frequency estimate is

```
> sum(a, na.rm = T)/(2 * sum(!is.na(a)))
```

```
[1] 0.116799
```

exactly the same as above.

The frequencies of the reference allele are computed very simply as one minus the frequency of the effective allele.

B.22 Exercise 27:

To test for HWE in first 10 SNPs in total sample

```
> summary(srdta@gtdata[, 1:10])
```

```
      NoMeasured CallRate        Q.2 P.11 P.12 P.22       Pexact         Fmax
rs10        2384   0.9536 0.13255034 1792  552   40 7.897327e-01 -0.006880004
rs18        2385   0.9540 0.28029350 1232  969  184 7.608230e-01 -0.007017332
rs29        2374   0.9496 0.13774221 1763  568   43 7.955141e-01 -0.007241148
rs65        2378   0.9512 0.71972246  182  969 1227 6.475412e-01 -0.010016746
rs73        2385   0.9540 0.01341719 2331   44   10 1.792470e-12  0.303150234
rs114       2393   0.9572 0.11679900 1868  491   34 7.663683e-01  0.005487764
rs128       2391   0.9564 0.02488499 2281  101    9 9.408599e-06  0.129600629
rs130       2379   0.9516 0.69377890  222 1013 1144 9.615127e-01 -0.002140946
rs143       2377   0.9508 0.47728229  655 1175  547 6.512540e-01  0.009313705
rs150       2369   0.9476 0.65998312  267 1077 1025 5.518478e-01 -0.012948436
              Plrt Chromosome
rs10  7.355343e-01          1
rs18  7.315304e-01          1
rs29  7.227853e-01          1
rs65  6.246577e-01          1
rs73  1.281197e-12          1
rs114 7.894076e-01          1
rs128 1.000431e-05          1
rs130 9.168114e-01          1
rs143 6.497695e-01          1
rs150 5.281254e-01          1
```

To test it in cases

```
> summary(srdta@gtdata[srdta@phdata$bt == 1, 1:10])
```

```
      NoMeasured  CallRate        Q.2 P.11 P.12 P.22       Pexact         Fmax
rs10        1197 0.9622186 0.13700919  888  290   19 4.635677e-01 -0.024514202
rs18        1189 0.9557878 0.28511354  605  490   94 7.759191e-01 -0.010949158
rs29        1176 0.9453376 0.14285714  859  298   19 2.832575e-01 -0.034722222
rs65        1185 0.9525723 0.72700422   83  481  621 4.647357e-01 -0.022595469
rs73        1187 0.9541801 0.01053075 1167   15    5 3.988770e-08  0.393614304
rs114       1190 0.9565916 0.12184874  918  254   18 8.924018e-01  0.002606831
```

```
rs128       1183 0.9509646 0.02409129 1129   51    3 2.747904e-02  0.083175674
rs130       1188 0.9549839 0.68392256  117  517  554 8.407527e-01 -0.006569292
rs143       1192 0.9581994 0.48489933  320  588  284 6.848365e-01  0.012522119
rs150       1182 0.9501608 0.66624365  127  535  520 5.568363e-01 -0.017756050
              Plrt Chromosome
rs10  3.871421e-01          1
rs18  7.052930e-01          1
rs29  2.214580e-01          1
rs65  4.348023e-01          1
rs73  2.423624e-08          1
rs114 9.285104e-01          1
rs128 3.157174e-02          1
rs130 8.207476e-01          1
rs143 6.654994e-01          1
rs150 5.409408e-01          1
```

in controls

```
> summary(srdta@gtdata[srdta@phdata$bt == 0, 1:10])

      NoMeasured  CallRate        Q.2 P.11 P.12 P.22       Pexact         Fmax
rs10        1177 0.9453815 0.12744265  897  260   20 7.933317e-01  0.006751055
rs18        1185 0.9518072 0.27426160  623  474   88 9.418133e-01 -0.004812165
rs29        1188 0.9542169 0.13215488  897  268   23 5.288436e-01  0.016525913
rs65        1183 0.9502008 0.71344041   98  482  603 8.871139e-01  0.003540522
rs73        1188 0.9542169 0.01641414 1154   29    5 6.941219e-06  0.244001185
rs114       1192 0.9574297 0.11157718  941  236   15 8.846527e-01  0.001356081
rs128       1197 0.9614458 0.02589808 1141   50    6 7.745807e-05  0.172107564
rs130       1181 0.9485944 0.70491109  104  489  588 8.887439e-01  0.004728114
rs143       1174 0.9429719 0.46805792  334  581  259 8.604122e-01  0.006165442
rs150       1176 0.9445783 0.65306122  139  538  499 7.968462e-01 -0.009574142
              Plrt Chromosome
rs10  8.178295e-01          1
rs18  8.683219e-01          1
rs29  5.737373e-01          1
rs65  9.031273e-01          1
rs73  5.537568e-06          1
rs114 9.627084e-01          1
rs128 7.552399e-05          1
rs130 8.710047e-01          1
rs143 8.326938e-01          1
rs150 7.424986e-01          1
```

Index