ABEL tutorial

Yurii Aulchenko

August 24, 2009

Contents

Chapter 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.

Originally GenABEL was developed to facilitate GWA analysis of quantitative traits using data coming from extended families and/or collected form genetically isolated populations. At the same time GenABEL implements a large number of procedures used in analysis of population-based data; it supports analysis of binary and quantitative tarits, and of survival (time-till-event) data. Most up-to-date information about GenABEL can be found at the web-site http://mga.bionet.nsc.ru/nlru/GenABEL/.

GenABEL is a part of more extensive ABEL collection (http://mga.bionet.nsc.ru/~yurii/ABEL/) of software supporting different kinds of GWA analyses.

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 ("5").

Otherwise, you can start with R basics and simple association analyises using few SNPs in section 2, "2". In the next section, 4 ("4") 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 ("5"), 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 ("6") is dedicated to analysis in presence of populational stratification and analysis of family-based data.

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

The last section, 11 ("11"), 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, "4").

Chapter 2

Introduction to R

In this section we will consider base R data types and operations, and tools for analysis of qualitative and quantitative traits. Only basic R functionality – the things which are crucial to know before we can proceed to genetic association analysis – will be covered within this section. If you want to make most of your data, though, we strongly recommend that you improve your knowledge of R using books other than this. A number of excellent manuals ('An introduction to R', 'Simple R', 'Practical Regression and Anova using R', and others) is available free of charge from the R project web-site (http://www.r-project.org).

In the first part of this chapter you will learn about the most important R data types and will learn how to work with R data. Next, we will cover exploratory data analysis. The chapter will end with introduction to regression analysis.

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 command line-based. When you first start R, a command prompt appears. To get help and overview of R, type `help.start()` on the command line and press `enter`. This will start internet browser and open the main page of the R documentation.

Let us first use R as a powerful calculator. You can directly operate with numbers in R. Try multiplying two by three:

```
> 2 * 3
```

```
[1] 6
```

Other standard arithmetic operations can be performed in similar manner:

```
> 2/3
```

```
[1] 0.6666667
```

(division)

```
> 2^3
```

```
[1] 8
```

(power)

```
> 2 - 3
```

```
[1] -1
```

(subtraction)

```
> 2 + 3
```

```
[1] 5
```

(summation)[1].

Mathematical functions, such as square roots, base-10 logarithm, and exponentiation, are available in R as well:

```
> sqrt(5)
```

```
[1] 2.236068
```

```
> log10(2.24)
```

```
[1] 0.350248
```

```
> exp(0.35)
```

```
[1] 1.419068
```

Here, we have computed e to the power of base-10 logarithm of the square root of the sum of two and three. After each operation performed, we have rounded the result to the two digits after the floating point – just in order to do less typing.

The arithmetic operations and functions can be nested and therefore we can obtain the above result in one line, and without the 2nd-digit approximation:

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

```
[1] 1.418337
```

R functions include not only the standard mathematical ones, but also a wide range of statistical function, for example, probability density functions of many probability distributions. We will make extensive use of these at a later stage, when computing significance and estimating statistical power.

For any function with name say '`fun`', help may be obtained by typing '`help(fun)`' on the command line.

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

Exercise 1. Explore the help page for the Wilcoxon test and answer the questions:

[1] For complete list of arithmetic operations try `help("+")`

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

If you do not know the exact name for the function you look for, try '`help.search("query")`', where `query` is the keyword.

Exercise 2. Try to find out what are the functions to do

1. Fisher exact test
2. T-test

One of important R operations is *assignment*, which is done with '`<-`' operator. A (new) variable name should be provided on the left-hand side of this operator and on the right-hand side, there must be either name of already existing variable or an expression. For example, we if want to assign value '2' to variable '`a`', and value '3' to the variable '`b`' we would use the assignment operator:

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

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

```
> b

[1] 3
```

Evaluation of the expression

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

[1] 1.418337
```

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

While the variables 'a' and 'b' contain single numeric values, variables in general can be multi-dimensional; an one-dimensional example of such is a vector (array). Let us create an example vector and experiment with it:

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

Here, '`c()`' is a function, which **c**ombines its arguments to make a vector. This vector is then assigned to a variable named '`v`'.

Now, let us try different operations with this vector:

```
> v + 1

[1]  2  4  6  8 12
```

It is easy to see that the result is a vector, which is obtained by adding one to each element of the original vector `v`. Other arithmetic operations and mathematical functions behave in the same way, e.g. the operation is performed for each element of the vector, and the results are returned:

```
> 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
```

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

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

and add it to the vector `v`:

```
> v + ov

[1]  2  5  8 11 16
```

You can see that the summation was done element-wise, i.e. the first element of the result vector is obtained as the sum of the first elements of `v` and `ov`, the second is the sum of the second elements, and so forth.

Other arithmetic operations with two vectors are performed in the same element-wise manner:

```
> v * ov

[1]  1  6 15 28 55
```

(multiplication)

```
> v^ov

[1]      1      9    125   2401 161051
```

(power).

The vector operations considered above returned a same-length vector as output. There are others – statistical and summary – functions which evaluate a vector as a whole and 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.:

```
> length(v)

[1] 5

> mean(v)

[1] 5.4
```

```
> var(v)

[1] 14.8
```

One of the basic, and probably most used, data operations in R is *sub-setting*. This refers to an 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 a bigger sub-set, 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)` stays for 'combine numbers from 2 to 4 into a vector'. An equivalent result is obtained by

```
> a[c(2, 3, 4)]

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

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

```
> 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 a sub-set based on some condition. Let us consider two vectors, `tmphgt`, containing the height of some subjects, and `tmpids`, containing their identification codes (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 need to derive the IDs of the people with height over 170 cm. To do that, we need to combine several operations. First, we shoudl run the logical function `>170` on the height data:

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

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

This returns a logical vector whose elements are '`TRUE`', when a particular element of the `tmphgt` satisfies the condition `>170`. The returned logical vector, in turn, can be applied to sub-set any other vector of the same length[2], including itself. Thus if you need to see what are the heights in people, which are taller than 170 cm, you can use

```
> tmphgt[vec]
```

```
[1] 175 182 173 192
```

As you can see, only the elements of `tmphgt`, for which the corresponding value of `vec` was '`TRUE`', are returned. In the same manner, the logical vector `vec` can be applied to select elements of the vector of IDs:

```
> 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 can use

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

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

```
> tmpids[vec]
```

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

A better[3] way to do logical sub-setting assumes use of the `which()` function on the top of the logical vector. This function reports which elements are `TRUE`. To obtain above results you can run:

[2] 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.

[3] Because it treats NAs for you

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

[1] 2 3 4

> tmpids[vec]

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

You can see that no `vec` contains a vector, whose elements are the indexes of the elements of `tmphgt` for which the logical condition satisfies.

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

     [,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, you can use

```
> a[2, 2]

[1] 22
```

To access the elemnt from the second row and third column, use

```
> a[2, 3]

[1] 32
```

Note that here, the row index (2) comes first, and the column index (3) comes second.

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 vector is returned for this dimension. For example, here we retrieve the first row

	1	2	3
1	1	4	7
2	2	5	8
3	3	6	9

Table 2.1: Vector representation of a matrix. Elements in the table are the vector indices of the matrix elements.

```
> a[1, ]

[1] 11 21 31
```

...and the 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
```

Other way to address elements of a matrix is to use one-dimensional index. For example, if you want to access element in the 2nd row and 2nd column, instead of

```
> a[2, 2]

[1] 22
```

you can use

```
> a[5]

[1] 22
```

This way of accessing the elements of a matrix is based on the fact, that each matrix can be preseted as a vector, whose elements are numbered consequtively: the element in the upper-left corner has index 1, the element in the second row of the first column has index 2, and the last elemnt in the borrom-right corner has the maximal value, as shown in Table 2.1.

As well as with vectors, you can sub-set matrices using logical conditions or indexes. 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, better

```
> which(a > 21)

[1] 5 6 7 8 9
```

Note that in the latter case, a vector whose elements give the 1-D indexes of the matrix, is returned. This vector indicates the elemnts of matrix `a`, for which the condition `(a>21)` is satisfied.

You can obtain the values of the matrix's elements, for which the condition isfulfilled, either by

```
> a[a > 21]

[1] 22 23 31 32 33
```

or

```
> a[which(a > 21)]

[1] 22 23 31 32 33
```

Once again, the latter method should be prefered. Consider an example, where some elements of the matrix are missing (`NA`) – a situation which is common in real data analysis. Let us replace the elemnt number 5 with `NA` and perform sub-setting operations on the resulting matrix:

```
> a

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

> a[5] <- NA
> a

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

> a[a > 21]

[1] NA 23 31 32 33

> a[which(a > 21)]

[1] 23 31 32 33
```

You can see that when `a[a>21]` was used, not only the elements which are greater than 21 were returned, but also `NA` was. As a rule, this is not what you want, and `which` should be used unless you do want to make some use of the `NA` elements.

In this section, we have generated a number of R data objects. Some of these were numeric (e.g. vector of heights, `tmphgt`) and some were character, or string (e.g. vector of study IDs, `tmpids`). Some times you need to figure out what is the class of a certain object. This can be done using the `class()` function. For example,

```
> tmphgt

[1] 150 175 182 173 192 168

> class(tmphgt)

[1] "numeric"

> tmpids

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

> class(tmpids)

[1] "character"
```

What happens if we try to find out the class of

```
> a

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

– an object, which contains a matrix?

```
> class(a)

[1] "matrix"
```

Results are expected – we find out that `a` is a matrix, which is correct. At the same time, a matrix is an upper-level class, which contains a number of elemnts, belonging to some lower-level (e.g. character/numeric) class. To see what is the class of the matrix's elements, try

```
> a[1, ]

[1] 11 21 31

> class(a[1, ])

[1] "numeric"
```

which says that elemnts (at least of the first row) are numeric. Because all elements of a matrix should have the same class, we can conclude that `a` is a matrix containing numeric values.

At this point, it is worthwile inspecting what data objects were created during our work. This can be done with the `ls()` command:

```
> ls()
[1] "a"      "b"      "dje"    "ov"     "tmphgt" "tmpids" "v"      "vec"
```

Obviously, this "list" command is very useful – you will soon find that it is just too easy to forget the name of a variable which it took long time to create. Some times you may wish to remove some of the data objects because you do not need then anymore. You can remove an object using the `rm()` command, where the names of objects to be deleted are listed as arguments. For example, to remove `tmphgt` and `tmpids` variable you can use

```
> rm(tmphgt, tmpids)
```

If you now look up what data obejcts are still left in you workspace with the `ls()` command

```
> ls()
[1] "a"   "b"   "dje" "ov"  "v"   "vec"
```

you find that you have successfully deleted `tmphgt` and `tmpids`.

At this point, you can exit R by typing `q()` on the command line and pressing `Enter`.

Summary:

- You can get access to the top-level R documentation by `help.start()` command. To search help for some keyword `keywrd`, you can use `help.search(keywrd)` command. To get description of some function `fun`, use `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`
- You can see objects available in your workspace by using the `ls()` command
- Unnecessary object (say, `tmphgt`) can be deleted from the workspace using `rm` command, e.g. `rm(tmphgt)`
- You can leave R using the `q()` command

Exercise 1.

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

A *data frame* is a class of R data, which, basically, is a data table. In such tables, it is usually assumed that rows correspond to subjects (observations) and columns correspond to variables (characteristics) measured on these subjects. A nice feature of data frames is that columns (variables) have names, and the data can be addressed by referencing to these names[4].

We will explore R data frames using example data set `assoc`. Start R with double-click on the file named `assocbase.RData`. You can see the names of the loaded objects by using the "list" command:

```
> ls()
```

```
[1] "assoc"
```

Thus, only one object is loaded. The class of this object is:

```
> class(assoc)
```

```
[1] "data.frame"
```

– a data frame.

The dimensionality of a data frame (or a matrix) can be determined by using the `dim()` command:

```
> dim(assoc)
```

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

```
[1] 250   7
```

Here, the first number corresponds to the number of rows (subjects) and the second to the number of columns (variables). Thus, the data frame `assoc` contains the data on 250 subjects, who are characterised by 7 variables each.

Let us now figure out what are the names of the 7 variables present in the data frame. To see what are the variable names, use the command `names()`:

```
> names(assoc)
```

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

These variables correspond to the personal identifier (ID, variable `subj`), sex, affection status, quantitative trait `qt` and several SNPs. Each variable can have its own type (numeric, character, logic), but all variables must have the same length – thus forming a matrix-like data structure.

A variable from a data frame (say, `fram`), which has some name (say, `nam`) can be accessed through `fram$nam`. This will return a conventional vector, containing the values of the variable. 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  1  0  0  1  0
 [26]  0  1  0  0  1  1  0  0  0  0  0  0  0  0  1  0  0  1 NA  0  0  0  0  0  0
 [51]  1  1  0  1  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0
 [76]  0  0  1  1  1  0  0  0  0  0  0  0  0  1  1  0  0  0  0  1 NA  0  0  0  0
[101]  1  0  0  1  1  0  1  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  1  0
[126]  1  0  0  0  1  0  0  0  1  0  1 NA  1  0  0  0  0  0  0  0  0  0  1  0  0
[151]  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
[176]  0  0  1  1 NA  0  0  0  0  1  1  1  1  0 NA  1  0  0  0  0  0  0  1  1  0
[201]  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  1  0  0  0  0  0
[226]  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
```

The `aff` (affected) variable here codes for a case/control status, conventionally, the cases are coded as "1" and controls as "0". You can also see several "NA"s, which stays for missing observation.

Exercise 2.

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

Data frame may be thought of as a matrix which is a collection of (potentily different-type) vectors. All sub-setting operations discussed before for matrices are applicable to a data frame, while all operations dicussed for vectors are applicable to data frame's variables.

Thus, as any particular variable present in a data frame is a conventional vector, its elements can be accessed using the vector's indices. For example, if you would like to know what are the ID, sex and affection status for the person with index 75, you can request

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

```
[1] 1409
```

```
> assoc$sex[75]

[1] 1

> assoc$aff[75]

[1] 0
```

Alternatively, using the matrix-style of sub-setting, you can see all the data for person 75:

```
> assoc[75, ]

   subj sex aff       qt snp4 snp5 snp6
75 1409   1   0 1.014664  A/B  B/A  B/B
```

In the same manner as with matrices, you can get data for e.g. subjects 5 to 15 by

```
> assoc[5:15, ]

   subj sex aff         qt snp4 snp5 snp6
5  1533   0   0  0.1009220  A/B  B/A  B/A
6  2466   1   0 -0.1724321  A/B  A/A  A/A
7  2425   0   0 -0.3378473  B/B  A/A  A/A
8  1068   0   0 -1.7112925  A/A  B/B <NA>
9   198   1   0 -0.4815822  A/B  B/A  B/A
10 1496   1   0  1.2281232  A/A  B/B  B/B
11  909   0   0  0.5993945  A/B  B/A  B/A
12 1213   0   0  1.9792190  A/A  B/B  B/B
13  181   1   0  1.5435921  A/A  B/B  B/B
14 1783   0   0 -1.6242738  A/B  B/A  B/A
15 1914   0   0 -0.5160331  A/A  B/B  B/B
```

The result is actually a new data frame containing data only on people with index from 5 to 15:

```
> x <- assoc[5:15, ]
> class(x)

[1] "data.frame"

> dim(x)

[1] 11  7
```

As well as with matrices and vectors, it is possible to sub-set elements of a data frame based on (a combination of) logical conditions. For example, if you are interested in people who have the `qt` values over 1.4, you can find out what are the indices of these people

```
> 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
```

and then show the compelte data with

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

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

At the same time, if you only want to check what are the IDs of these people, try

```
> 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 find what are the IDs and what are the SNP genotypes of these people, we can try

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

```
    subj snp4 snp5 snp6
12  1213  A/A  B/B  B/B
13   181  A/A  B/B  B/B
33  1737  A/A  B/B  B/B
41  1319  A/A  B/A  B/A
54   516  A/B  B/A  B/A
68  1355  A/A  B/B  B/B
72   186  A/A  B/A  B/A
76  1426  A/B  B/A  B/A
89  1284  A/A  B/B  B/B
106  822  A/B  B/A  B/A
118 2129  A/B  B/A  B/A
142  212  A/B  B/A  B/A
156 1443  A/A  B/B  B/B
161  704  A/B  B/A  B/A
175 1648  A/B  B/A  B/A
181 1628  A/B  B/A  B/A
193  562  A/A  B/B  B/B
219  858  A/B  B/A  B/A
241  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).

The same result can be obtained using variables' names insted of the variables' indices. To remind you the variables' names:

```
> names(assoc)
```

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

And now make a vector of the variables' names of interest and filter the data based on it:

```
> namstoshow <- c("subj", "snp4", "snp5", "snp6")
> assoc[vec, namstoshow]
```

```
    subj snp4 snp5 snp6
12  1213  A/A  B/B  B/B
13   181  A/A  B/B  B/B
33  1737  A/A  B/B  B/B
41  1319  A/A  B/A  B/A
54   516  A/B  B/A  B/A
68  1355  A/A  B/B  B/B
72   186  A/A  B/A  B/A
76  1426  A/B  B/A  B/A
89  1284  A/A  B/B  B/B
106  822  A/B  B/A  B/A
118 2129  A/B  B/A  B/A
142  212  A/B  B/A  B/A
156 1443  A/A  B/B  B/B
161  704  A/B  B/A  B/A
175 1648  A/B  B/A  B/A
181 1628  A/B  B/A  B/A
193  562  A/A  B/B  B/B
219  858  A/B  B/A  B/A
241  698  B/B  A/A  A/A
```

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.

```
> subj[75]
```

```
[1] 1409
```

instead of `assoc$subj[75]`.

While it is possible to explore the data presented in a data frame using the sub-setting operations and screen output, and modify certain data elements using the assignment ("`<-`") operation, you can also explore and modify the data contained in a data frame[5] by using `fix()` command (e.g. try `fix(assoc)`). However, normally this is not necessary.

With attached data frames, a possible complication is that later on you may have several data frames which contain the variables with the same names. The variable which will be used when you directly use the name would be the one from the data frame attached last. You can use `detach()` function to remove a certain data frame from the search path, e.g. after

```
> detach(assoc)
```

we can not use direct reference to the name (try `subj[75]`) anymore, but have to use the full path instead:

[5] and also a matrix

```
> assoc$subj[75]

[1] 1409
```

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 3.

Load the `srdta` data object supplied with GenABEL by loading the package with `library(GenABEL)` and then loading the data with `data(srdta)`. The `srdta` object 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 the 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 first variable is?
3. What is the sum of variable 2? Can you guess what data variable 2 contains?

2.3 Exploratory analysis of qualitative and quantitative traits

Let us now attach the data frame `asscoc`

```
> attach(assoc)
```

and explore it.

Let us first check how many of the subjects are males. In the `sex` variable, males are coded with "1" and females with "0". Therefore to see the numer of males, you can use

```
> sum(sex == 1)

[1] 129
```

and to determine what is male sex proportion you can use

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

[1] 0.516
```

This way to compute the proportion would only work correctly if there are no missing observations (`lenght()` returns the total length of a variable, including `NA`s).

Because of the way the males are coded, the same answer is reached by

```
> mean(sex)

[1] 0.516
```

However, that would not have worked if the sex was coded differently, e.g. with "1" for males and "2" for females.

Let us now try to find out the mean of the quantitative trait `qt`. By definition, the mean of a variable, say x (with i-th element denoted as x_i) is

$$\bar{x} = \frac{\Sigma_{i=1}^{N} x_i}{N}$$

where N is the number of measurements.

If we try to find out the mean of `qt` by direct use of this formula, we first need to find out the sum of the `qt`'s elements. The `sum()` function of R precisely does the operation we need. However, if we try it

```
> sum(qt)

[1] NA
```

this returns "NA". The problem is that the `qt` variable contains "NA"s (try `qt` to see these) and by default the "NA" is returned. We can, however, instruct the `sum()` function to remove "NA"s from consideration:

```
> sum(qt, na.rm = T)

[1] -26.4733
```

where `na.rm=T` tells R that missing variables should be be removed (NonAvailable.ReMove=True)[6].

We can now try to compute the mean with

```
> sum(qt, na.rm = T)/length(qt)

[1] -0.1058932
```

This result, however, is not correct. The `length()` function returns the total length of a vector, which includes "NA"s as well. Thus we need to compute the number of the `qt`'s elements, which are not missing.

For this, we can use R function `is.na()`. This function returns `TRUE` if supplied argument is missing (`NA`) and `FALSE` otherwise. Let us apply this function to the vector `assoc$qt`:

[6] The same argument works for a number of R statistical functions such as `mean`, `median`, `var`, etc

```
> is.na(qt)

  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
 [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [37]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
 [49] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
[133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[157] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Indeed, the 7 missing elements are correctly identified. However, we are interested in elements which are **not** missing. To get these, we can use the logical function NOT (!), which changes all `FALSE` to `TRUE` and visa versa:

```
> !is.na(qt)

  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
 [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [37] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
 [49]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [73]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [97]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[109]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[145]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[157]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[169]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[181]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[193]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
[205]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[217]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[229]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[241]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Thus the number of elements which are not missing[7] is

```
> sum(!is.na(qt))

[1] 243
```

Finally, we can compute the mean of the `qt` with

```
> sum(qt, na.rm = T)/sum(!is.na(qt))

[1] -0.1089436
```

While this way of computing the mean is enlightening in the sense of how to treat the missing values, the same correct result should be normally achieved by supplying the `na.rm=T` argument to the `mean()` function:

```
> mean(qt, na.rm = T)

[1] -0.1089436
```

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

```
> table(sex)

sex
  0   1
121 129
```

which, again, tells us that there are 129 males and 121 females in this data set. This function excludes missing observations form consideration.

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

As with arithmetic operations and mathematical functions, most of the R operations can be combined within a single line. Let us try to combine logical conditions and the `table()` command to check the distribution of number of affected in men and women separately:

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

 0  1
95 31

> table(aff[which(sex == 0)])

 0  1
95 24
```

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

[7] A hidden trick here is that arithmetic operations treat `TRUE` as one and `FALSE` as zero

Exercise 4.

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 (how many different genotype classes are present and what are the counts)?

Contingency tables for pairs of variables (cross-tables) can be generated in R using the `table` command we have used in previous section 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 95 24
  1 95 31
```

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

As is usually the 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)
> class(a)

[1] "table"

> a

   aff
sex  0  1
  0 95 24
  1 95 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 0.38775510 0.09795918
  1 0.38775510 0.12653061
```

Needless to say, this is equivalent to

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

             0          1
  0 0.38775510 0.09795918
  1 0.38775510 0.12653061
```

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 0.5000000 0.4363636
  1 0.5000000 0.5636364
```

saying us that 56.4% 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 0.7983193 0.2016807
  1 0.7539683 0.2460317
```

saying us that 56.4% 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 = 0.4457
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.676759 2.482869
sample estimates:
odds ratio
  1.290313
```

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  68
  1   8  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 0.7948718 0.7719298 0.8000000
  1 0.2051282 0.2280702 0.2000000
```

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 = 0.2511, df = 2, p-value = 0.882
```

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 5.

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)`**.**

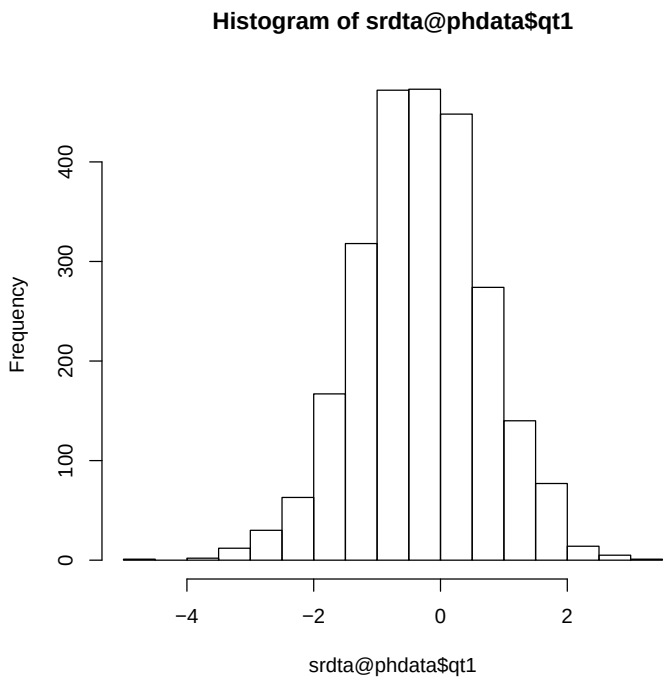

Figure 2.1: Histogram of qt1

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 2.1)

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 2.2)

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`.

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)
```

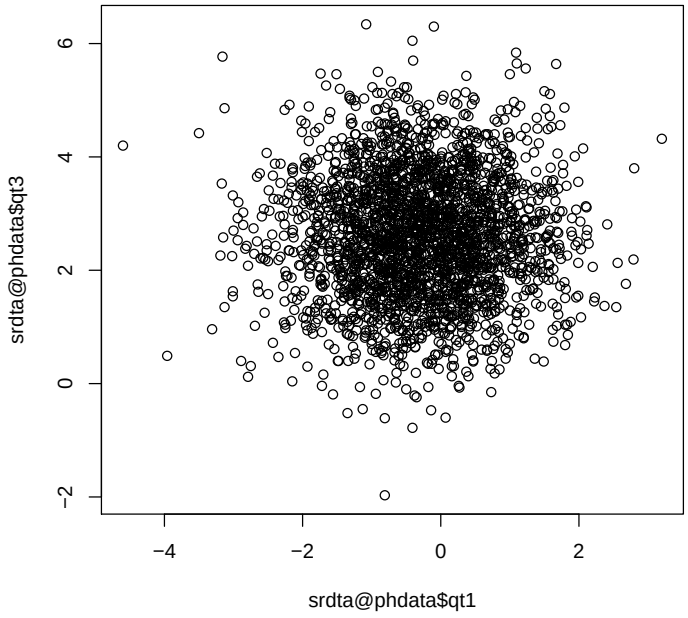

Figure 2.2: Scatter-plot of qt1 against qt3

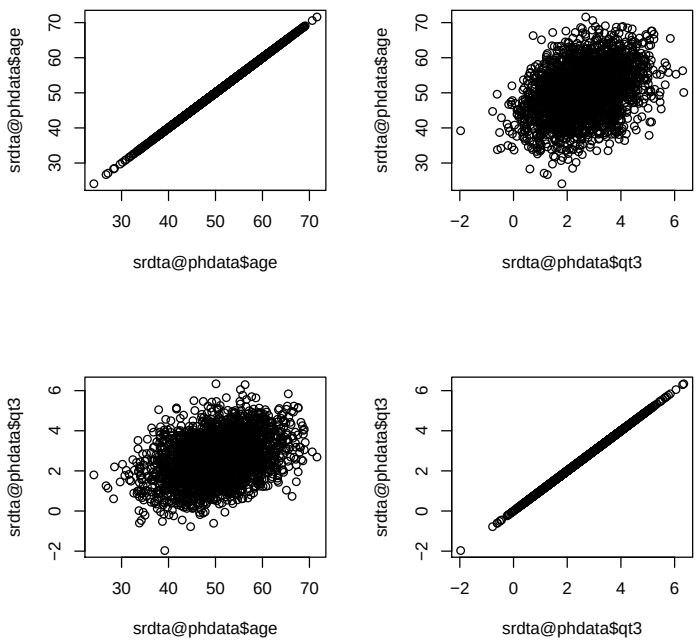

Figure 2.3: Quality control graph for `sex, age, qt3`

```
--------------------------------
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 6.

Explore `phdata` slot of `srdta`

1. How many people has age over 65 years?
2. What is the sex distribution (proportion of males) in the people over 65 years old?

Exercise 7.

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?

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 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 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
```

Chapter 3

Introduction to genetic 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 Characterisation of genetic data

3.2 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 18). 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 20:

```
> 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

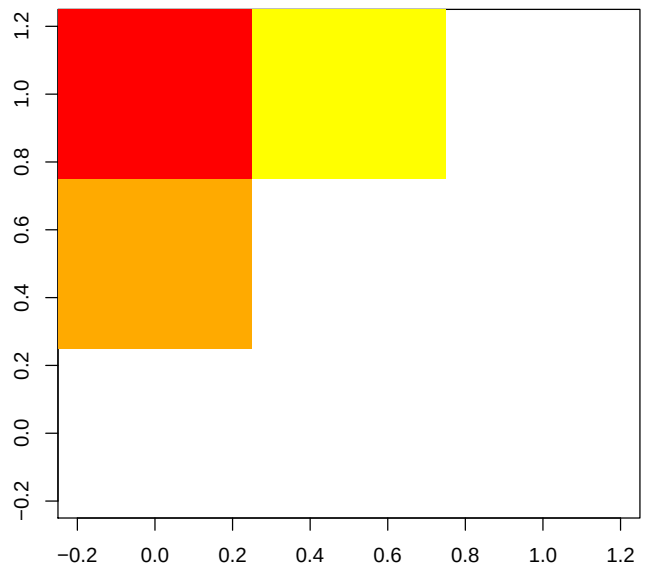

Figure 3.1: r^2 plot for snp4, snp5 and snp6

```
> 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 3.1.

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
```

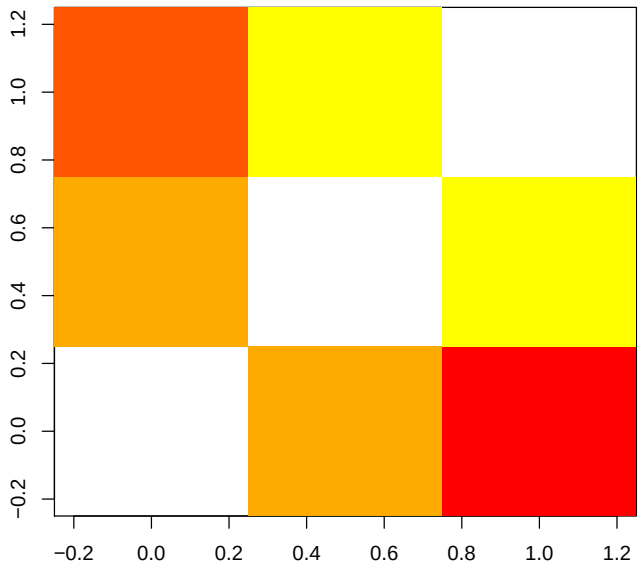

Figure 3.2: r^2 plot for snp4, snp5 and snp6. Above diagonal: LD in cases; below: controls

```
> 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
```

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 3.2):

```
> 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)
```

3.3 Genetic association analysis

3.4 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

```
> 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 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 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 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 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.

3.5 Exercise

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

Exercise 1 *How many SNPs are described in this data frame?*

Exercise 2 *What is the frequency (proportion) of snp1 allele A? What is its frequency in these affected (aff==1)?*

Exercise 3 *How many cases and controls are present?*

Exercise 4 *If all subjects are used to test HWE, are there any SNPs out of HWE at nominal $P \leq 0.05$? Which ones?*

Exercise 5 *If only controls are used to test the SNPs which are out of HWE in total sample, are these still out of HWE?*

Exercise 6 *Which SNP pairs are in strong LD ($r^2 \geq 0.8$)?*

Exercise 7 *For SNPs in strong LD, what is r^2 for separate samples of cases and controls?*

Exercise 8 *Is there significant association between affection status and sex? What is P-value for association?*

Exercise 9 *Is association between the disease and `qt` significant?*

Exercise 10 *Which SNPs are associated with the quantitative trait `qt` at nominal $P \leq 0.05$? Use 2 d.f. test.*

Exercise 11 *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 12 *For the SNPs selected in previous question, test association using additive model. Which SNPs are still associated?*

Exercise 13 *If you adjust the analysis under additive model (question 12) for significant covariates which you discovered in questions 8 and 9, are these findings still significant?*

Exercise 14 *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?*

Chapter 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 4.1.

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

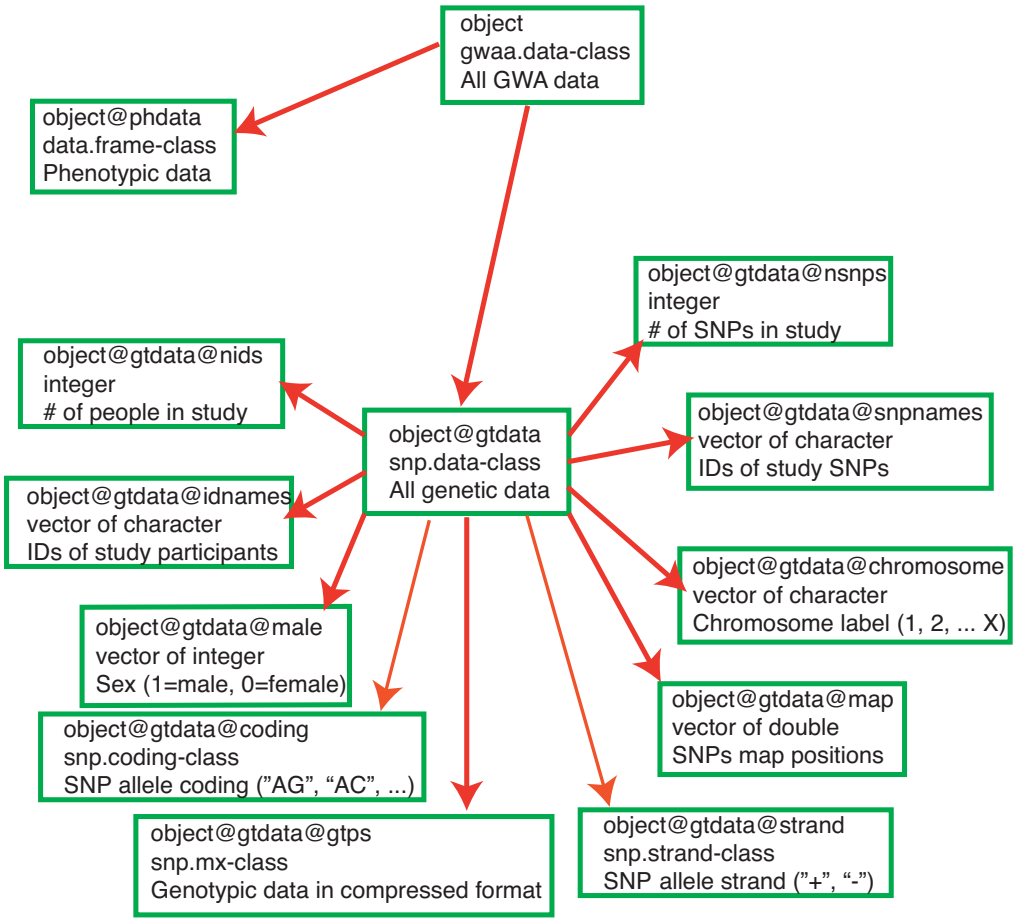

Figure 4.1: Structure of gwaa.data-class. In every box, first line contains the object and slot names, second line describes the class of this object, and third line describes what information is contained.

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 4.1). 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 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 8.

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 9.

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 10.

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 sub-setting 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 11.

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)

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 4.2).

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
```

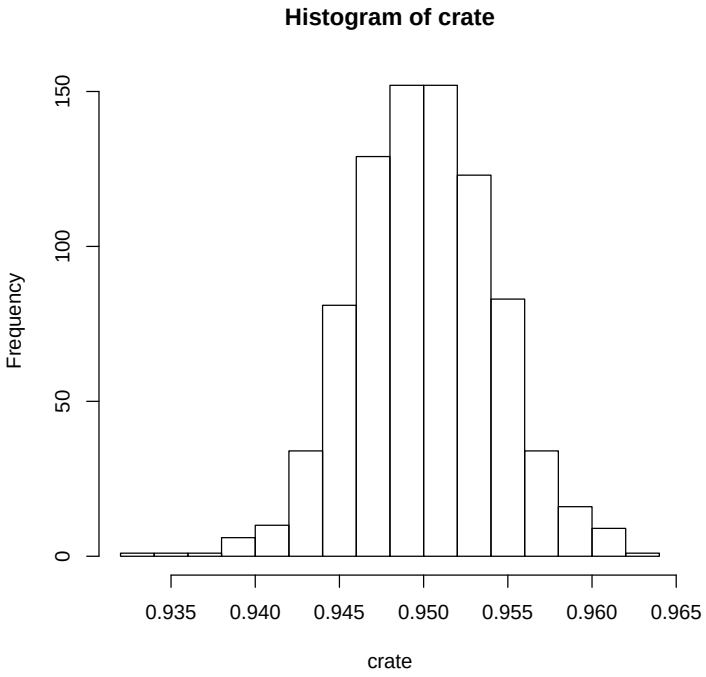

Figure 4.2: Histogram of the call rate

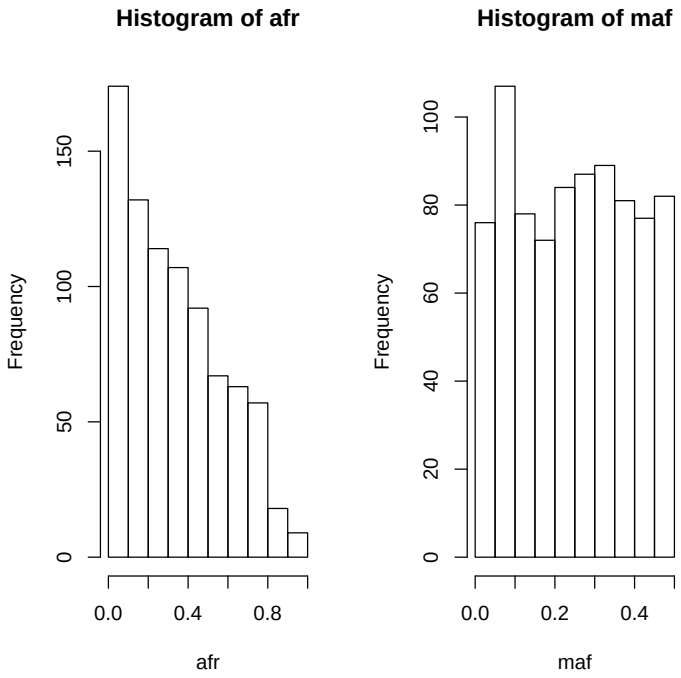

Figure 4.3: Histogram of the call rate

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 4.3) 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, ])

    NoMeasured       Hom    E(Hom)       Var           F    CallPP       Het
p1         707 0.7751061 0.6201205 0.4048662  0.40798616 0.8487395 0.2248939
p2         714 0.7198880 0.6212102 0.5090002  0.26050805 0.8571429 0.2801120
p3         700 0.5757143 0.6217225 0.4332890 -0.12162558 0.8403361 0.4242857
p4         705 0.5602837 0.6196322 0.5251900 -0.15602916 0.8463385 0.4397163
p5         707 0.6265912 0.6222997 0.5288936  0.01136232 0.8487395 0.3734088
p6         703 0.7382646 0.6218624 0.3770418  0.30783027 0.8439376 0.2617354
p7         709 0.5867419 0.6178574 0.4527349 -0.08142388 0.8511405 0.4132581
p8         711 0.6680731 0.6193635 0.5163296  0.12796887 0.8535414 0.3319269
p9         711 0.6315049 0.6202163 0.5599395  0.02972375 0.8535414 0.3684951
p10        713 0.6185133 0.6215808 0.4889042 -0.00810600 0.8559424 0.3814867
```

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)
```

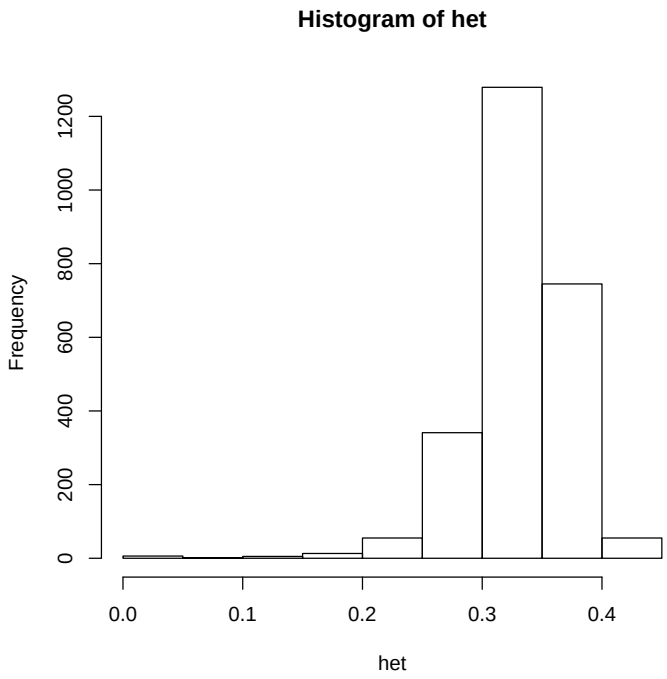

Figure 4.4: Histogram of heterozygosity

The resulting histogram is presented in figure 4.4. 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.

Exercise 12.

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

Chapter 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=0)    Mean     SD No(by.var=1)    Mean     SD   Ptt   Pkw
id               50      NA     NA           86      NA     NA    NA    NA
sex              50   0.420  0.499           86   0.593  0.494 0.053 0.052
age              50  47.038 13.971           86  50.250 12.206 0.179 0.205
dm2              50      NA     NA           86      NA     NA    NA    NA
height           49 167.671  8.586           86 170.448 10.362 0.097 0.141
weight           49  76.534 17.441           86  93.587 27.337 0.000 0.000
diet             50   0.060  0.240           86   0.058  0.235 0.965 0.965
bmi              49  27.304  6.463           86  32.008  8.441 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    0.012    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       59.000       76.000      0
Prop  0.007           0        0.434        0.559      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] 0.2487219

$`Standard deviation of mean heterozygosity for a person`
[1] 0.04309648
```

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 1522.00
Prop   0.058        0.169       0.168      0.224    0.38

$`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   0.024     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           50            0      0
Prop      0           0            1            0      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] 0.2601554

$`Standard deviation of mean heterozygosity for a person`
[1] 0.04859507
```

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

100100

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

```
> 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] 1.029658

$se
[1] 0.0005343007

$iz0
[1] 1

$iz2
[1] 1
```

The estimate of λ is 1.03, suggesting inflation of the test and some degree of stratification.

The λ is computed by regression in a Q-Q plot. Both estimation of λ and production of the $\chi^2 - \chi^2$ plot can be done using the `estlambda` function:

```
> estlambda(an0$P1df)

$estimate
[1] 1.029658

$se
[1] 0.0005343007
```

The corresponding $\chi^2 - \chi^2$ plot is presented in Figure 5.1.

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

We can also present the obtained results using the "Manhatten plot", where a SNP genomic position is on the X-axes and $-log_{10}$ of the p-value is shown on Y-axes:

```
> plot(an0)
```

The resulting plot is presented in the figure 5.2. By default, $-log_{10}(P-value)$ of not corrected 1 d.f. test 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 = c("lightblue", "lightgreen"))
```

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

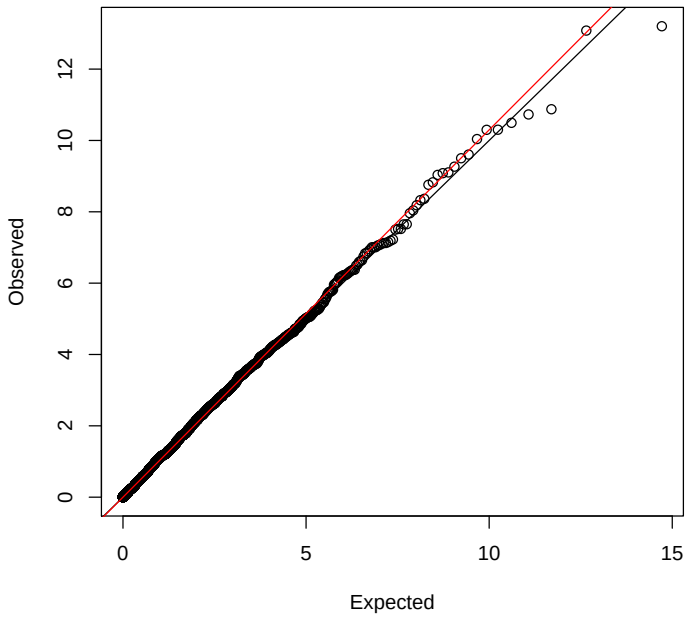

Figure 5.1: $\chi^2 - \chi^2$ plot for a GWA scan. Black line of slope 1: expected under no inflation; Red line: fitted slope.

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

```
          Chromosome Position   N       effB         P1df        Pc1df
rs1719133          1  4495479 136 0.33729339 0.0002795623 0.0003425105
rs2975760          3 10518480 134 3.80380024 0.0002983731 0.0003649107
rs7418878          1  2808520 136 3.08123060 0.0009743183 0.0011540593
rs5308595          3 10543128 133 3.98254950 0.0010544366 0.0012463049
rs4804634          1  2807417 132 0.43411456 0.0011970132 0.0014100096
rs3224311          2  6009769 135 3.15831710 0.0013290907 0.0015611949
rs26325            3 10617781 135 0.09742793 0.0013313876 0.0015638203
rs8835506          2  6010852 132 3.17720829 0.0015321522 0.0017928676
rs3925525          2  6008501 135 2.98416931 0.0019400358 0.0022558582
rs2521089          3 10487652 135 2.50239493 0.0020524092 0.0023829357
              effAB      effBB         P2df
rs1719133 0.4004237  0.0000000 0.0006333052
rs2975760 3.4545455 10.0000000 0.0011434877
rs7418878 3.6051282  4.8717949 0.0022642036
rs5308595 3.3171429        Inf 0.0045930101
rs4804634 0.5240642  0.1739130 0.0036964462
rs3224311 3.4151786  4.2500000 0.0029405999
rs26325   0.1097724         NA 0.0013313876
rs8835506 3.4903846  4.1250000 0.0031618340
```

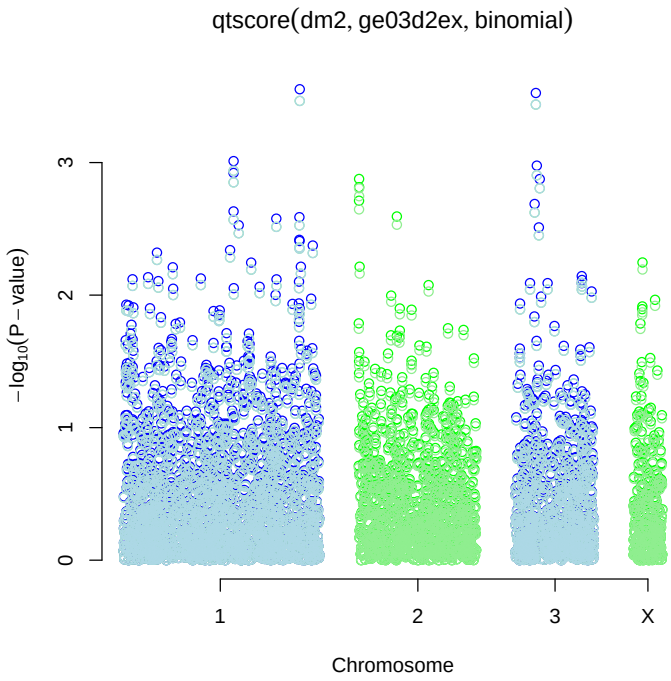

Figure 5.2: $-log_{10}(P-value)$ from the genome scan before QC procedure. Raw analysis: black; corrected analysis: green

```
rs3925525 3.2380952  4.1212121 0.0045554384
rs2521089 2.5717703  4.7727273 0.0069661425
```

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 `qtscore()` function with `times` argument, which tells it to compute empirical GW (or experiment-wise) significance

```
> an0.e <- qtscore(dm2, ge03d2ex, times = 200, quiet = TRUE)
```

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

```
          Chromosome Position   N       effB  P1df Pc1df      effAB      effBB
rs1719133          1  4495479 136 -0.2652064 0.425 0.510 -0.2080882 -0.7375000
rs2975760          3 10518480 134  0.2340655 0.460 0.525  0.2755102  0.4090909
rs7418878          1  2808520 136  0.2089098 0.835 0.900  0.2807405  0.3268398
```

```
rs5308595          3 10543128 133  0.2445516 0.865 0.925  0.2564832  0.4623656
rs4804634          1  2807417 132 -0.2050449 0.910 0.945 -0.1193830 -0.3845238
rs3224311          2  6009769 135  0.2133633 0.940 0.950  0.2778634  0.3151515
rs26325            3 10617781 135 -0.4875367 0.940 0.955 -0.4875367         NA
rs8835506          2  6010852 132  0.2112000 0.950 0.965  0.2796221  0.3076923
rs3925525          2  6008501 135  0.2057095 0.970 0.985  0.2660834  0.3074627
rs2521089          3 10487652 135  0.1775016 0.975 0.985  0.2254633  0.3396072
          P2df
rs1719133 9.99
rs2975760 9.99
rs7418878 9.99
rs5308595 9.99
rs4804634 9.99
rs3224311 9.99
rs26325   9.99
rs8835506 9.99
rs3925525 9.99
rs2521089 9.99
```

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 0.7732328 (s.e. 0.02044899), 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, 129 (96.26866%) people passed all criteria

RUN 2
3653 markers and 129 people in total
72 (1.970983%) markers excluded as having low (<1.937984%) 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 0.2744972 (s.e. 0.01706096)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7711307 (s.e. 0.01884635), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 3581 (98.02902%) markers passed all criteria
In total, 129 (100%) people passed all criteria

RUN 3
3581 markers and 129 people in total
0 (0%) markers excluded as having low (<1.937984%) 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 0.2744972 (s.e. 0.01706096)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
```

```
Mean IBS is 0.7690901 (s.e. 0.01819136), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 3581 (100%) markers passed all criteria
In total, 129 (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   376     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 129 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   65.000 121.000 240.000  3581
Prop    0.012    0.018   0.034   0.067     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    70.00 127.000 228.000  3581
Prop    0.013     0.02   0.035   0.064     1
```

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

```
> 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  0.496562976 3262.00000000 3261.00000000 3249.00000000 3265.0000000
id300 -0.012267995    0.49486164 3268.00000000 3257.00000000 3271.0000000
id403 -0.012464681   -0.01262302    0.51836209 3255.00000000 3270.0000000
id415 -0.002399026    0.01363777   -0.02528089    0.56733776 3259.0000000
id666 -0.019116287   -0.02110468    0.02096914   -0.02025885    0.5684808
```

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 5.3. 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] "id2097" "id6954" "id2136" "id858"

> cl2

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

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.

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...
3581 markers and 125 people in total
0 people excluded because of call rate < 0.1
0 markers excluded because of call rate < 0.1
Passed: 3581 markers and 125 people

Running sex chromosome checks...
```

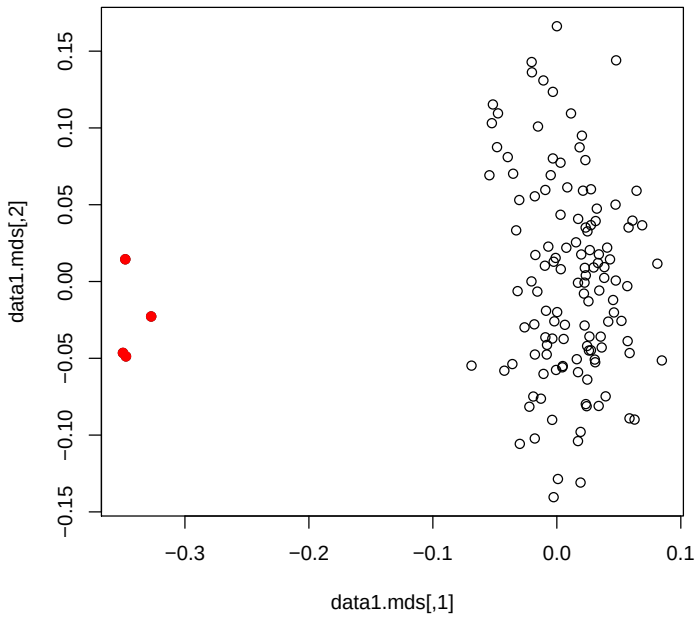

Figure 5.3: Mapping samples on the space of the first two Principle Components resulting from analysis of genomic kinship. Red dots identify genetic outliers

```
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: 3581 markers and 125 people

no X/Y/mtDNA-errors to fix


RUN 1
3581 markers and 125 people in total
40 (1.117006%) markers excluded as having low (<2%) 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 0.2776868 (s.e. 0.01655360)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7720158 (s.e. 0.01258361), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 3541 (98.883%) markers passed all criteria
In total, 125 (100%) people passed all criteria
```

```
RUN 2
3541 markers and 125 people in total
0 (0%) markers excluded as having low (<2%) 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 0.2776868 (s.e. 0.01655360)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.772 (s.e. 0.01237184), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 3541 (100%) markers passed all criteria
In total, 125 (100%) people passed all criteria

> summary(qc2)

$`Per-SNP fails statistics`
          NoCall NoMAF NoHWE Redundant Xsnpfail
NoCall         0     0     0         0        0
NoMAF         NA    40     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       0        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=0)    Mean     SD No(by.var=1)    Mean     SD   Ptt   Pkw
id               48      NA     NA           77      NA     NA    NA    NA
sex              48   0.438  0.501           77   0.597  0.494 0.084 0.082
age              48  46.378 13.865           77  50.593 12.465 0.089 0.097
dm2              48      NA     NA           77      NA     NA    NA    NA
height           47 167.988  8.610           77 170.423 10.646 0.166 0.223
weight           47  77.273 17.427           77  94.160 26.963 0.000 0.000
diet             48   0.062  0.245           77   0.065  0.248 0.957 0.957
bmi              47  27.485  6.539           77  32.235  8.335 0.001 0.001
       Pexact
id         NA
sex     0.098
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  22.000  108.00  3541
Prop        0    0.001   0.006    0.03     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] 1.040969

$se
[1] 0.0007325815

$iz0
```

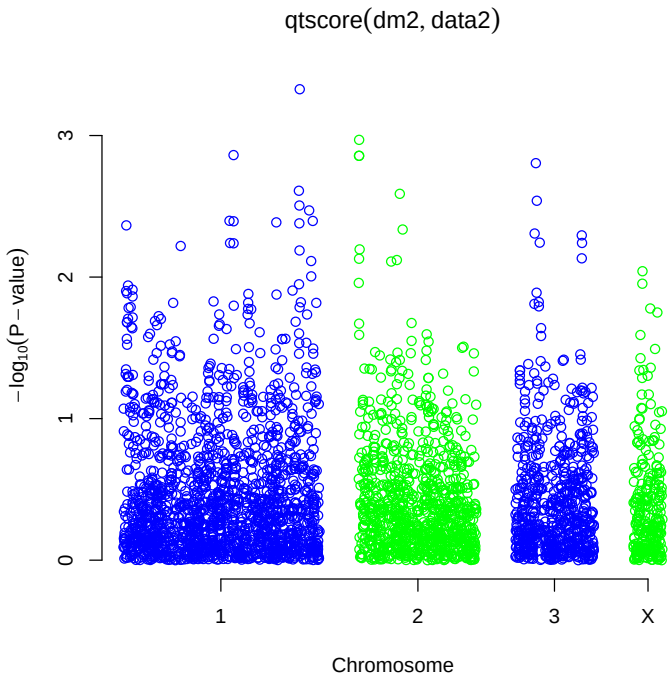

Figure 5.4: $-log_{10}(CorrectedP - value)$ from the genome scan after the QC procedure

```
[1] 1.016119

$iz2
[1] 1
```

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 5.4).

Produce the scan summary by

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

```
          Chromosome Position   N       effB         P1df       Pc1df
rs1719133          1  4495479 125 -0.2709715 0.0003599745 0.000470699
rs8835506          2  6010852 122  0.2346446 0.0008465818 0.001072548
rs4804634          1  2807417 122 -0.2161383 0.0010949169 0.001374104
rs3925525          2  6008501 125  0.2280847 0.0011081904 0.001390146
rs3224311          2  6009769 125  0.2280847 0.0011081904 0.001390146
rs2975760          3 10518480 124  0.2244898 0.0012545339 0.001566573
rs4534929          1  4474374 124 -0.1864322 0.0019996882 0.002454805
```

```
rs6079246          2  7048058 124 -0.4730491 0.0021060217 0.002580453
rs5308595          3 10543128 123  0.2731788 0.0023674963 0.002888472
rs1013473          1  4487262 125  0.1864494 0.0025659783 0.003121460
               effAB      effBB         P2df
rs1719133 -0.2153891 -0.7260274 0.0009290908
rs8835506  0.3201507  0.3201507 0.0012972329
rs4804634 -0.0971660 -0.3947368 0.0026493765
rs3925525  0.3046200  0.3187614 0.0020102814
rs3224311  0.3046200  0.3187614 0.0020102814
rs2975760  0.2570423  0.3959311 0.0047945922
rs4534929 -0.1487634 -0.3968254 0.0074296050
rs6079246 -0.4730491         NA 0.0021060217
rs5308595  0.2844101  0.4719101 0.0095505795
rs1013473  0.2714833  0.3782654 0.0067821374
```

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 <- qtscore(dm2, data2, times = 200, quiet = TRUE)
```

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

```
          Chromosome Position   N        effB  P1df Pc1df       effAB
rs1719133          1  4495479 125 -0.27097152 0.480 0.570 -0.21538910
rs8835506          2  6010852 122  0.23464458 0.775 0.845  0.32015066
rs4804634          1  2807417 122 -0.21613833 0.850 0.925 -0.09716599
rs3925525          2  6008501 125  0.22808468 0.860 0.925  0.30461997
rs3224311          2  6009769 125  0.22808468 0.860 0.925  0.30461997
rs2975760          3 10518480 124  0.22448980 0.885 0.935  0.25704225
rs7435137          1  4259040 124  0.08916655 1.000 1.000  0.11073664
rs664063           2  7288020 125 -0.13014211 1.000 1.000 -0.09180036
rs546570           2  6120257 124 -0.15740741 1.000 1.000  0.75000000
rs7908680          1  2311762 122  0.19316239 1.000 1.000  0.19316239
               effBB  P2df
rs1719133 -0.7260274 0.710
rs8835506  0.3201507 0.785
rs4804634 -0.3947368 0.965
rs3925525  0.3187614 0.935
rs3224311  0.3187614 0.935
rs2975760  0.3959311 0.995
rs7435137  0.1761787 1.000
rs664063  -0.6372549 1.000
rs546570   0.5925926 1.000
rs7908680         NA 1.000
```

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 <- qtscore(dm2 ~ sex + age, data2, times = 200, quiet = T)
```

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

```
          Chromosome Position   N        effB  P1df Pc1df       effAB
rs1719133          1  4495479 125 -0.25890157 0.570 0.690 -0.22773582
rs8835506          2  6010852 122  0.22215111 0.905 0.970  0.31550163
rs3925525          2  6008501 125  0.21871930 0.930 0.975  0.30498571
rs3224311          2  6009769 125  0.21871930 0.930 0.975  0.30498571
rs4804634          1  2807417 122 -0.20651613 0.955 0.985 -0.09071397
rs6079246          2  7048058 124 -0.46876373 0.985 0.995 -0.46876373
rs7435137          1  4259040 124  0.06938366 1.000 1.000  0.10716530
rs664063           2  7288020 125 -0.11264399 1.000 1.000 -0.07332386
rs546570           2  6120257 124 -0.15868138 1.000 1.000  0.13634042
rs7908680          1  2311762 122  0.10136925 1.000 1.000  0.10136925
              effBB  P2df
rs1719133 -0.62102151 0.915
rs8835506  0.28148154 0.825
rs3925525  0.28255981 0.900
rs3224311  0.28255981 0.900
rs4804634 -0.37649275 0.995
rs6079246          NA 1.000
rs7435137  0.13499367 1.000
rs664063  -0.61187789 1.000
rs546570  -0.02234096 1.000
rs7908680          NA 1.000
```

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 <- qtscore(dm2 ~ sex + age, data2, ids = ((bmi > 30 &
+     dm2 == 1) | dm2 == 0), times = 200, quiet = TRUE)
```

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

```
          Chromosome Position  N       effB  P1df Pc1df      effAB      effBB
rs1891586          1  2297398 89 -0.2525526 0.860 0.870 -0.1844922 -0.4744112
rs794264           1  2534738 88  0.2285093 0.945 0.960  0.3184520  0.4947580
rs9178808          1  2536431 89 -0.2265653 0.970 0.970 -0.1652287 -0.4983534
rs5032886          1  2537020 89 -0.2265653 0.970 0.970 -0.1652287 -0.4983534
rs9630764          1  3897972 89  0.2231745 0.975 0.980  0.2110261  0.4420613
rs7504607          1  2704056 89 -0.4582995 0.975 0.980 -0.4582995         NA
rs2884479          X 13618173 87 -0.2085201 0.990 0.990 -0.4488573 -0.3298415
rs4803085          1  3256810 89  0.2793141 0.995 0.995  0.2658124  0.6359811
```

```
rs3215698          X 13559835 89 -0.2679440 0.995 0.995 -0.3703538 -0.4693541
rs8538767          X 13437421 87  0.2328469 0.995 0.995  0.2120678  0.4763338
           P2df
rs1891586 0.995
rs794264  1.000
rs9178808 1.000
rs5032886 1.000
rs9630764 1.000
rs7504607 1.000
rs2884479 0.410
rs4803085 1.000
rs3215698 1.000
rs8538767 1.000
```

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

Exercise 16 *How many markers are presented in the original data set?*

Exercise 17 *Is there evidence for inflation of the HWE test staistics?*

Exercise 18 *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 19 *How many male turned apparently female?*

Exercise 20 *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 21 *How many "twin" DNAs did you discover?*

Exercise 22 *How many genetic outliers did you discover?*

After you have finished QC, answer the questions:

Exercise 23 *How many cases and controls are presented in the data after QC?*

Exercise 24 *How many markers are presented in the data after QC?*

Exercise 25 *Is there evidence for inflation of the HWE test staistics?*

Exercise 26 *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 27 *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 28 *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 29 *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?

Chapter 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 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 0 0 0 0
 [75] 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 0 0 0 0 0
[112] 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 = c("lightgreen", "lightblue"), cex = 1.2)
```

The resulting plot is presented at figure 6.1.

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:

```
> tmp.gkin <- data1.gkin
> diag(tmp.gkin) <- hom(data1[, autosomal(data1)])$Var
> data1.eg <- egscore(dm2, data = data1, kin = tmp.gkin)
```

Note that we have replaced the diagonal elements of the genomic kinship matrix, which are by default are 0.5+ th estimates of the inbreeding, by the variance returned by the `hom` function. The analysis plot may be added to the previous one by

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

The resulting plot is presented at figure 6.1.

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 13.

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*?)

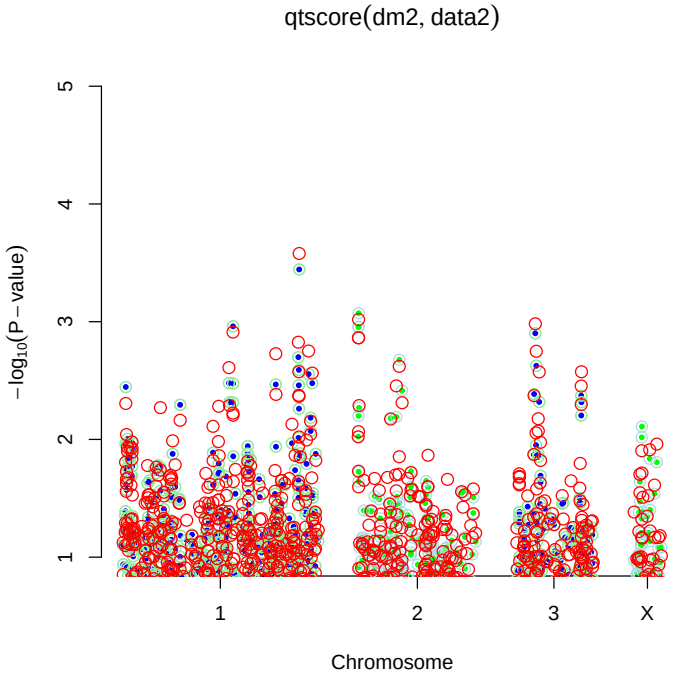

Figure 6.1: Comparison of structured association analysis (green), method of Price et al. (red) and analysis excluding genetic outliers (solid black).

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

In this section we will consider analysis of quantitative traits in a family-based cohort, where participants were not selected for the value of the trait under analysis. Such data may be generated in any study selecting participants based on kinship (e.g. collections of sibships, nuclear or extended families); also any study in a genetically isolated population is likely to end up with a large proportion of relatively closely related individuals, even if ascertainment was random with the respect to kinship.

In pedigree-based association analysis the pedigree works as a confounder – exactly in the same manner as ethnic origin may work in a population-based study. Any genetic polymorphism is inherited through genealogy, and therefore

genotypes are more similar between close relatives. In the same manner, any other heritable trait will be also more similar between relatives, and therefore certain degree of association is expected between *any* genetic marker and *any* heritable trait in a family-based sample. If additive 1 d.f. test for association is considered, the effect of confounding by pedigree can be shown to inflate the resulting null distribution of presumably χ^2_1 test statistics by a certain constant λ.

As you remember, this is exactly what happens when simple test for association is applied to a population-based data with ethnic admixture. In a population-based study with strong admixture (both in terms of the proportion and ethnic "distance"), some genomic regions may have been differentially selected in different populations. In such situation, use of genomic control does not prevent false-positive association between a trait and these regions, and other methods, such as EIGENSTRAT or Structured Association, are to be used.

For pedigree-based data coming from (relatively) genetically homogeneous population it can be shown that λ is a function of trait's heritability and pedigree structure, expressed as kinship matrix. Thus, genomic control is a simple and valid method to study association in genetically homogeneous families. However, this method reduces (or summarises if you prefer) all the abundant information about heritability and relationship into a single parameter λ, therefore it is not the most powerful method.

In quantitative genetics, a mixed polygenic model of inheritance may be considered as "industrial standard" – this model has sound theoretical bases and is proven by time to describe well inheritance of complex quantitative traits. This model describes the vector of observed quantitative traits as

$$Y = \mu + G + e \tag{6.1}$$

where μ is the intercept, G is contribution from polygene, and e is random residual.

It is assumed that for each individual its "personal" random residual e_i is distributed as Normal with mean zero and variance σ^2_e. As these residuals are independent between pedigree members, the joint distribution of residuals in the pedigree can be modelled using multivariate normal distribution with variance-covariance matrix proportional to the identity matrix I (this is a matrix with diagonal elements equal to 1, and off-diagonal elements equal to zero): $e \sim MVN(0, I\sigma^2_e)$.

The polygenic component G describes the contribution from multiple independently segregating genes all having a small additive effect onto the trait (infinitesimal model). For a person for whom parents are not known, it is assumed that G_i is distributed as Normal with mean zero and variance σ^2_G. Assuming model of infinitely large number of genes, it can be shown that given polygenic values for parents, the distribution of polygene in offspring follows Normal distribution with mean $(G_m + G_f)/2$ and variance $\sigma^2_G/2$, where G_m is maternal and G_f is paternal polygenic values. From this, it can be shown that jointly the distribution of polygenic component in a pedigree can be described as multivariate normal with variance-covariance matrix proportional to the relationship matrix Φ: $G \sim MVN(0, \Phi\sigma^2_G)$.

Thus the log-likelihood for this model can be written as a function of three

parameters:

$$L(\mu, \sigma_G^2, \sigma_e^2) = -\frac{1}{2} \cdot log_e \left|(\Phi \cdot \sigma_G^2 + I \cdot \sigma_e^2)\right| + (Y-\mu)^T \cdot (\Phi \cdot \sigma_G^2 + I \cdot \sigma_e^2)^{-1} \cdot (Y-\mu) \quad (6.2)$$

where μ is intercept, σ_G^2 is the proportion of variance explained by the polygenic component, and σ_e^2 is the residual variance.

Covariates such as sex, age, or a genetic marker studied for association can be easily included into the model:

$$Y = \mu + \sum_j \beta_j \cdot C_j + G + e$$

Here, C_j is the vector containing j-th covariate and $beta_j$ is the coefficient of regression of Y onto that covariate.

This mixed morel leads to likleihood

$$L(\mu, \sigma_G^2, \sigma_e^2, \beta_1, \beta_2, ...) = -\frac{1}{2} \cdot log_e \left|(\Phi \cdot \sigma_G^2 + I \cdot \sigma_e^2)\right| + \left(Y - (\mu + \sum_j \beta_j \cdot C_j)\right)^T \cdot \left(\Phi \cdot \sigma_G^2 + I \cdot \sigma_e^2\right)^{-1} \cdot \left(Y - (\mu + \sum_j \beta_j \cdot C_j)\right) \quad (6.3)$$

This general formulation can be easily adopted to test genetic association; for example, an effect of a SNP can be incorporated into regression model

$$Y = \mu + \beta_g \cdot g + G + e$$

where g is the vector containing genotypic values. In this mode, you can specify a variety of 1 d.f. models by different coding of the vector g. For example, if you consider an "AG" polymorphism and want to estimate and test additive effect of the allele "G", you should code "AA" as 0 (zero), "AG" as 1 and "GG" as 2. Under this coding, the β_g will estimate additive contribution from the "G" allele. If you are willing to consider dominant model for G, you should code "AA" and "AG" as 0 and "GG" as 1. Recessive and over-dominant models can be specified in a similar manner. If, however, you want to estimate general 2 d.f. model, the specification should be different:

$$Y = \mu + \beta_a \cdot g + \beta_d \cdot I_{g=2} + G + e$$

where g is coded as 0, 1 or 2, exactly the same as in the additive model, and $I_{g=2}$ is the binary indicator which takes value of one when g is equal to 2 and zero otherwise. In this model, β_a will estimate the additive and β_d – the dominance effect. There may be other, alternative coding(s) allowing for essentially the same model, for example

$$Y = \mu + \beta_1 \cdot I_{g=1} + \beta_2 \cdot I_{g=2} + G + e$$

would estimate trait's deviation in these with $g = 1$ (β_1) and these with $g = 2$ (β_2) from the reference ($g = 0$).

The classical way to estimate mixed polygenic model and test for significance is Maximum Likelihood (ML) or Restricted ML (REML) using equation (6.3). However, when large pedigrees are analysed, ML/REML solution may take prohibitively long time, i.e. from minutes to hours for single SNP analysis, making study of hundreds of thousand of SNPs impossible. Therefore fast approximate tests were developed for the purposes of GWA association analysis in samples of relatives.

Here we will cover two of fast approximations available, FAmily-based Score Test for Association (FASTA, Chen & Abecasis, 2007) and Genome-wide Rapid Analysis using Mixed Models And Score test (GRAMMAS, Amin et al, 2007). Both tests are based on the classical polygenic mixed model and are performed in two steps.

First, polygenic model as specified by equation (6.1) and likelihood (6.2) is estimated using available data.

Secondly, the maximum likelihood estimates (MLEs) of the intercept, $\hat{\mu}$, proportion of variance explained by the polygenic component, $\hat{\sigma}^2_G$, and residual variance, $\hat{\sigma}^2_e$, are used to compute the FASTA test statistics

$$T^2_F = \frac{\left((g - E[g])^T \cdot (\Phi \cdot \hat{\sigma}^2_G + I \cdot \hat{\sigma}^2_e)^{-1} \cdot (Y - \hat{\mu})\right)^2}{(g - E[g])^T \cdot (\Phi \cdot \hat{\sigma}^2_G + I \cdot \hat{\sigma}^2_e)^{-1} \cdot (g - E[g])}$$

It can be shown that T^2_F follows χ^2_1 when pedigree structure is 100% complete and 100% correct. As this is never actually the case, application of GC to correct for residual inflation is recommended.

FASTA test results in unbiased estimates of the SNP effect and correct $P - values$. Please keep in mind that this is correct – as for any score test – only when alternative is reasonably close to the null, i.e. when the SNP explains small proportion of trait's variance. Disadvantages of this test are that is can be relatively slow when thousands of study subjects are analysed, and that permutation procedures can not be applied to estimate genome-wide significance, because the data structure is not exchangeable.

Other test, GRAMMAS, also exploits MLEs from the polygenic model (6.1). However, these are used to first compute the vector of environmental residuals $\hat{e}$, using standard equation

$$\hat{e} = \hat{\sigma}^2_e \cdot (\Phi \cdot \hat{\sigma}^2_G + I \cdot \hat{\sigma}^2_e)^{-1} \cdot (Y - \hat{\mu})$$

These residuals, in turn, are used to run simple score test:

$$T^2_G = \frac{\left((g - E[g])^T \cdot \hat{e}\right)^2}{(g - E[g])^T \cdot (g - E[g])}$$

This test is conservative, but GC can be used to correct for the deflation of the test statistics.

The fact that environmental residuals $\hat{e}$ are not dependent on pedigree structure leads to a nice property of the GRAMMAS test: the data structure becomes exchangeable and permutations may be used to estimate genome-wide significance. When used in combination with GC, $P - values$ derived from

GRAMMAS test are correct; however, there is a downward bias in estimates of SNP effects.

When using FASTA or GRAMMAS test, it is recommended to estimate genomic kinship matrix from available genome-wide data, and use it in analysis instead of pedigree kinship. This solution firstly does not rely on the completeness and quality of pedigree; secondly, genomic kinship is more likely to give a better estimate of a *true* covariance between individual genomes, while pedigree kinship provides one with expectation. Therefore use of genomic kinship is expected to lead to better estimates of polygenic model, and thus better power to detect association in GWA analysis. This being said, we generally advocate use of genomic, and not pedigree kinship. Of cause, you can only implement this solution when you have GWA data; in a candidate gene study you will have to rely on the pedigree structure to estimate kinship matrix.

6.3 Example GWA analysis using family-based data

In this section, we will explore small data set (150 people, 5827 SNPs). Let us load and explore it:

```
> load("RData/erfsmall.RData")
> ls()

[1] "erfs"  "pkins"

> class(erfs)

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

> class(pkins)

[1] "matrix"
```

You can see that there a two objects, `erfs` and `pkins`, presented in the data. The class of the first object is standard `GenABEL` 's `gwaa.data-class`; this is the object containing GWA data. The other object contains kinship matrix, as estimated from pedigree data.

You can check the number of people and SNPs in the data set with

```
> erfs@gtdata@nids

[1] 150

> erfs@gtdata@nsnps

[1] 5827
```

As usual, it is advisable to check the distribution of SNPs by chromosome:

```
> table(erfs@gtdata@chromosome)
```

```
  1  10  11  12  13  14  15  16  17  18  19   2  20  21  22  23   3   4   5   6
484 251 224 285 188 210 206 189 159 170 149 481 130 100 132  26 403 300 320 397
  7   8   9   X
286 253 206 278
```

(here, 23 stays for pseudo-autosomal region of the X chromosome); you can see that markers are evenly spread over the chromosomes.

Summary marker statistics can be generated by

```
> descriptives.marker(erfs@gtdata)

$`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    17.000       26.000      75.000    437.000 5272.000
Prop   0.003        0.004       0.013      0.075    0.905

$`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          2   14.000  83.000 319.000  5827
Prop        0    0.002   0.014   0.055     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       1.000         3.00       14.000 132.00
Prop      0       0.007         0.02        0.093   0.88

$`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   77.000      33.000      214.000      208.000 5295.000
Prop  0.013       0.006        0.037        0.036    0.909

$`Mean heterozygosity for a SNP`
[1] 0.4402752

$`Standard deviation of the mean heterozygosity for a SNP`
[1] 0.08287253

$`Mean heterozygosity for a person`
[1] 0.4358498

$`Standard deviation of mean heterozygosity for a person`
[1] 0.01306670
```

You can see that the quality of genotypic data is quite reasonable: call rate is generally high, both per-person and per SNP, and there is little deviation from Hardy-Weinberg equilibrium.

Let us explore pedigree kinship matrix. First, let us just look how this matrix looks like by displaying few elements from the upper-left corner:

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

```
             id1         id2         id3         id4         id5
id1 5.00000e-01 8.56146e-05 1.01984e-04 2.33397e-04 8.56146e-05
id2 8.56146e-05 5.00000e-01 3.96513e-03 2.56896e-05 2.51269e-01
id3 1.01984e-04 3.96513e-03 5.00000e-01 1.21593e-05 3.96513e-03
id4 2.33397e-04 2.56896e-05 1.21593e-05 5.00000e-01 2.56896e-05
id5 8.56146e-05 2.51269e-01 3.96513e-03 2.56896e-05 5.00000e-01
```

By definition, pedigree kinship should take values between 0 and 0.5 (plus some small amount from inbreeding); kinship between (non-inbred) sibs or an offspring and the parent is 1/4. You can see that in the upper-left corner there is one inbred sib-pair (or parent-offspring pair; "id2" and "id5"). You can also see that this matrix is symmetric around the diagonal.

Let us summarise the distribution of kinship coefficients; in doing this we want to generate the summary for every off-diagonal element only once. Function `lower.tri` can be used to get the "lower triangle" sub-matrix elements:

```
> summary(pkins[lower.tri(pkins)])
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000000 0.0004999 0.0028500 0.0062810 0.0053780 0.2633000
```

As you can see, average relationship corresponds to that expected between second cousins ($1/64 = 0.015625$) and third cousins ($1/256 = 0.00390625$).

We can also draw a histogram of the distribution of the kinship coefficients (shown at figure 6.2A):

```
> hist(pkins[lower.tri(pkins)])
```

and see that most elations are indeed remote.

Let us estimate genomic kinship matrix using autosomal data with the command `ibs`, and look up the elements in the upper-left corner:

```
> gkins <- ibs(erfs[, autosomal(erfs)], weight = "freq")
> gkins[1:5, 1:5]
```

```
             id1           id2           id3           id4          id5
id1  0.531729506  5.439000e+03  5.446000e+03 5441.00000000 5440.0000000
id2 -0.012569446  4.921817e-01  5.524000e+03 5524.00000000 5521.0000000
id3  0.001516184 -8.551323e-03  5.144766e-01 5529.00000000 5528.0000000
id4  0.010459422 -1.471218e-02 -3.467894e-03    0.52042474 5523.0000000
id5 -0.007957596  2.561484e-01 -8.925127e-03   -0.02205665    0.4943998
```

Here, the estimated kinship is shown below the diagonal, and the number of informative SNP pairs used for estimation is shown above the diagonal.

You can see that "genomic kinship" coefficients may take values lower than zero, which is consequence of the fact that in effect "genomic kinship" is simply covariance between the vectors of individual genotypes. This quantity, though it provides an unbiased estimate of kinship, can be lower than zero.

```
> summary(gkins[lower.tri(gkins)])
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.038980 -0.011930 -0.005826 -0.003362  0.000481  0.268000
```

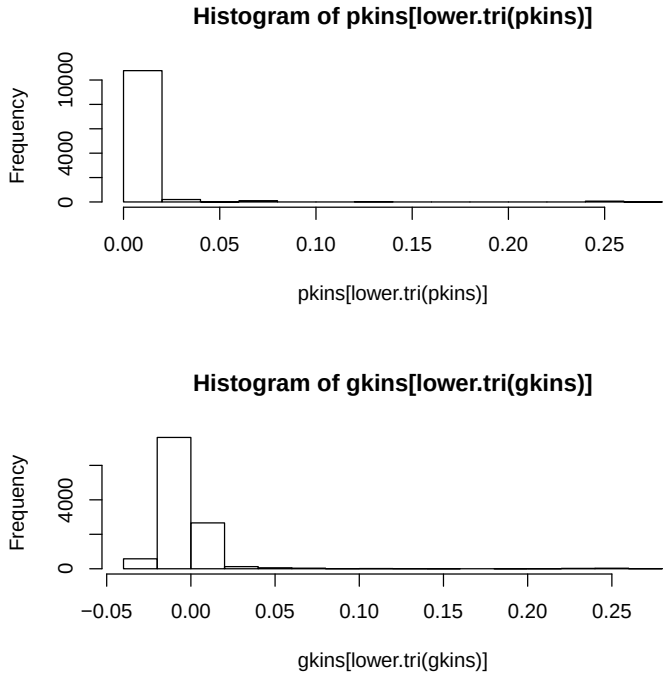

Figure 6.2: Distribution of the pedigree (upper histogram) and genomic (lower histogram) kinship coefficients for `erfs` data set.

here, the average is quite close to that obtained with pedigree kinship.

We can also draw a histogram of the distribution of "genomic kinship" coefficients (shown at figure 6.2B):

```
> hist(gkins[lower.tri(gkins)])
```

and can easily graphically present relations between genomic and pedigree kinship with

```
> plot(pkins[lower.tri(pkins)], gkins[lower.tri(gkins)])
```

(shown at figure 6.3), and estimate correlation between the two with

```
> cor(pkins[lower.tri(pkins)], gkins[lower.tri(gkins)])
```

```
[1] 0.91615
```

From the graph, you can clearly see that, though there is a very strong correlation between genomic and pedigree kinships, these are not identical.

In real data, you may find that there are some points where pedigree data clearly suggest relation different from that suggested by genomic data. Which one to believe? Generally, pedigrees are more prone to errors than genotypic data. In the data containing close relatives it is better to rely on "genomic kinship".

Let us first analyse the data using plain GC method:

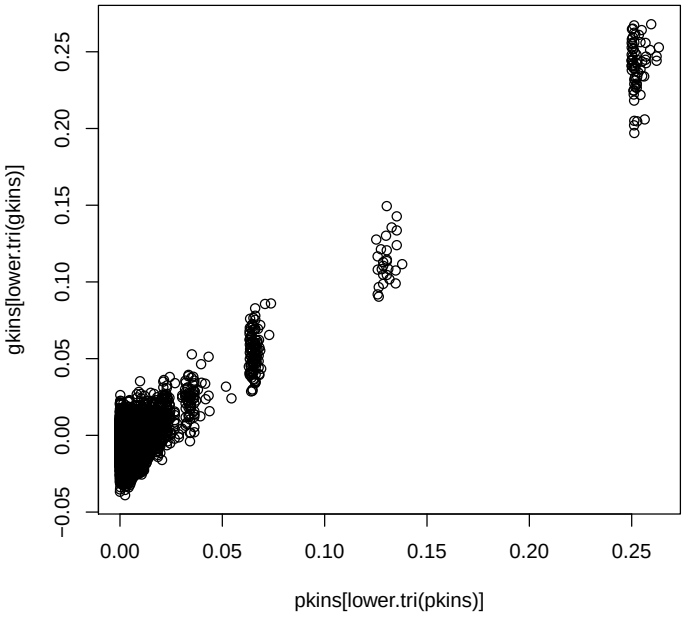

Figure 6.3: Scatter-plot relating pedigree and genomic kinships for `erfs` data set.

```
> qts <- qtscore(qtbas, data = erfs)
```

You can check the estimate of the inflation factor λ with

```
> qts$lam$est
```

```
[1] 1.137247
```

This is relatively high value, suggesting presence of close relatives in data and high heritability of the trait.

The top 10 hits from GWA analysis can be displayed with

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

```
           Chromosome   Position   N       effB         P1df        Pc1df
rs1781670         13 2694164735 150  0.4554848 0.0001804816 0.0004453466
rs1054889          2  324154337 150  0.4760959 0.0002711808 0.0006387800
rs1861659         16 3171591122 150  0.4844231 0.0002811280 0.0006594988
rs1860991          5 1213061568 150  0.4616985 0.0002834991 0.0006644252
rs1043883          8 1745120471 128  0.4338657 0.0005729702 0.0012396586
rs1872087         17 3329938025 150 -0.4111272 0.0006091602 0.0013088424
rs1982904          4  961162979 150  0.5369605 0.0006904958 0.0014626936
rs2141693         12 2590653716 150  1.2178255 0.0007070577 0.0014937647
rs7141672         14 2851979738 150  0.4595401 0.0007473279 0.0015689781
```

```
rs1075456         15 3036078968 150 -0.3943230 0.0008094764 0.0016841742
                effAB      effBB         P2df
rs1781670  0.55710736  0.9015131 0.0007574863
rs1054889  0.32660599  1.0001217 0.0008416365
rs1861659  0.30184684  1.0290361 0.0006959544
rs1860991  0.49412227  0.9080701 0.0013452354
rs1043883  0.23834763  0.9427557 0.0015609770
rs1872087 -0.50149169 -0.8092418 0.0024403173
rs1982904  0.60470761  0.8678924 0.0026214066
rs2141693  1.21782551         NA 0.0007070577
rs7141672  0.37923110  1.0172357 0.0028443093
rs1075456 -0.01486038 -0.8540990 0.0002978615
```

here, nominal $P - values$ after genomic control are given in column named "Pc1df".

We can estimate genome-wide empirical significance by using the same function with `times` argument, which tells the number of permutations:

```
> qts.e <- qtscore(qtbas, data = erfs, times = 200, quiet = TRUE)
```

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

```
          Chromosome   Position   N        effB  P1df Pc1df      effAB
rs1781670         13 2694164735 150  0.45548481 0.490 0.835  0.5571074
rs1054889          2  324154337 150  0.47609594 0.670 0.925  0.3266060
rs1860991          5 1213061568 150  0.46169852 0.685 0.930  0.4941223
rs1861659         16 3171591122 150  0.48442305 0.680 0.930  0.3018468
rs1982904          4  961162979 150  0.53696053 0.950 0.990  0.6047076
rs1043883          8 1745120471 128  0.43386568 0.900 0.990  0.2383476
rs1872087         17 3329938025 150 -0.41112719 0.915 0.990 -0.5014917
rs2017143          1    2252811 150  0.06610235 1.000 1.000  0.3084208
rs2840531          1    2318570 150 -0.31150314 1.000 1.000 -0.3273450
rs2477703          1    2458154 150 -0.16765976 1.000 1.000 -0.2043520
                effBB  P2df
rs1781670  0.90151313 0.935
rs1054889  1.00012170 0.945
rs1860991  0.90807011 0.995
rs1861659  1.02903607 0.915
rs1982904  0.86789245 1.000
rs1043883  0.94275573 1.000
rs1872087 -0.80924177 1.000
rs2017143  0.05342849 1.000
rs2840531 -0.47957726 1.000
rs2477703 -0.29647369 1.000
```

(argument "quiet" supress warning messages; this is used for the purposes of this tutorial, normally you do not need to specify this option)

As you can see, in this analysis nothing comes even close to genome-wide significance, as indicated by genome-wide corrected $P-values$ (column "Pc1df") all $>> 0.05$.

Let us try to estimate polygenic model with

```
> h2 <- polygenic(qtbas, kin = gkins, data = erfs)
```

The results of estimation are contained in "h2an" element of the resulting analysis object:

```
> h2$h2an

$minimum
[1] 139.9891

$estimate
[1] 0.03828507 0.78448170 1.21485857

$gradient
[1] -4.04701109 -0.06361344 -3.05067753

$code
[1] 2

$iterations
[1] 4
```

In the "estimate" list, the MLEs shown correspond to intercept $\hat{\mu}$, heritability $\hat{h}^2 = \hat{\sigma}^2_G/(\hat{\sigma}^2_G + \hat{\sigma}^2_e)$, and total variance $\hat{\sigma}^2_T = \hat{\sigma}^2_G + \hat{\sigma}^2_e$. You can see that heritability of the trait is indeed high – almost 80%.

Under these conditions (hight heritability, presence of close relatives) we may expect that FASTA and GRAMMAS analysis exploiting heritability model and relationship matrix in exact manner may have better power compared to simple GC.

Let us run FASTA test using estimated polygenic model, as specified by `h2` object:

```
> mms <- mmscore(h2, data = erfs)
```

There is little residual inflation left when we use "genomic kinship" matrix:

```
> mms$lam$est

[1] 1.022223
```

And the significance of "top" hit becomes an order of magnitude better compared to plain GC:

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

          Chromosome   Position   N       effB         P1df        Pc1df effAB
rs1075456         15 3036078968 150 -0.5384183 0.0000119801 0.0000149041    NA
rs1054889          2  324154337 150  0.4981468 0.0002076476 0.0002435181    NA
rs1781670         13 2694164735 150  0.4724096 0.0002398578 0.0002804606    NA
rs1264007          X 4256199578 150 -0.3772189 0.0007196424 0.0008227652    NA
rs774033          12 2531521679 150 -0.4283929 0.0009456322 0.0010751485    NA
rs7141672         14 2851979738 150  0.4605357 0.0010869289 0.0012323077    NA
rs537848          15 3020539910 150  0.4151429 0.0012151235 0.0013745365    NA
```

```
rs760109           X 4353106270 150  0.3677194 0.0013603032 0.0015352473    NA
rs1860991          5 1213061568 150  0.4111162 0.0014521083 0.0016366936    NA
rs1370139          6 1381958118 149  0.5911044 0.0016504575 0.0018554446    NA
          effBB P2df
rs1075456    NA 9.99
rs1054889    NA 9.99
rs1781670    NA 9.99
rs1264007    NA 9.99
rs774033     NA 9.99
rs7141672    NA 9.99
rs537848     NA 9.99
rs760109     NA 9.99
rs1860991    NA 9.99
rs1370139    NA 9.99
```

If you compare these results to that obtained with simple GC, you can also see that the ranks of top hits have changed quite a bit; unbiased estimated of genetic effects were obtained.

However, we can not estimate genome-wide significance with FASTA, because the data structure is not exchangeble.

Using GRAMMAS method, you can estimate nominal $P - values$ by

```
> grs <- qtscore(h2$pgres, data = erfs, clam = FALSE)
> grs$lam$est

[1] 0.797455
```

In the above analysis, note that the estimated "inflation" factor λ is less than one, i.e. now it is the GRAMMAS *deflation* factor. In order to obtain non-concervative test statistics, we had to say to qtscore that deflation is OK (parameter `clam=FALSE`).

We can see "top" nominal corrected $P - values$ with

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

          Chromosome   Position   N        effB         P1df        Pc1df
rs1075456         15 3036078968 150 -0.10704009 0.0001344706 0.0000190625
rs1054889          2  324154337 150  0.10245323 0.0009999196 0.0002288717
rs1781670         13 2694164735 150  0.09165369 0.0015542793 0.0003948600
rs1264007          X 4256199578 150 -0.07764670 0.0023993775 0.0006751315
rs774033          12 2531521679 150 -0.09282392 0.0026100856 0.0007490689
rs7141672         14 2851979738 150  0.09341129 0.0040026811 0.0012695452
rs537848          15 3020539910 150  0.08489591 0.0042136070 0.0013525421
rs1860991          5 1213061568 150  0.08655964 0.0042663779 0.0013734589
rs760109           X 4353106270 150  0.07432363 0.0045810469 0.0014994038
rs2639197         15 3068765468 150 -0.09085234 0.0052630062 0.0017791082
                effAB      effBB         P2df
rs1075456 -0.04882437 -0.2241217 0.0002410666
rs1054889  0.04392601  0.2236716 0.0013187233
rs1781670  0.08025377  0.1843682 0.0064411019
rs1264007 -0.09842481 -0.1512012 0.0099102406
```

```
rs774033  -0.09430729 -0.1856043 0.0107644457
rs7141672  0.08525171  0.1967954 0.0153935223
rs537848   0.08848437  0.1695158 0.0165978319
rs1860991  0.10807561  0.1629485 0.0140947764
rs760109   0.06139187  0.1491903 0.0179490427
rs2639197 -0.09197031 -0.1804026 0.0203630654
```

By comparing this output to that from FASTA test, you can see that $P-values$ are quite close, but the effects are underestimated with GRAMMAS, as expected.

However, the streangths of GRAMMAS test is not only its speed, but also possiblity to estimate genome-wide significance. This can be done by

```
> grs.e <- qtscore(h2$pgres, data = erfs, times = 200, clam = FALSE,
+     quiet = TRUE)



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

          Chromosome   Position   N         effB  P1df Pc1df       effAB
rs1075456         15 3036078968 150 -0.107040094 0.415 0.090 -0.04882437
rs1054889          2  324154337 150  0.102453231 0.990 0.630  0.04392601
rs1781670         13 2694164735 150  0.091653694 1.000 0.820  0.08025377
rs1264007          X 4256199578 150 -0.077646696 1.000 0.945 -0.09842481
rs774033          12 2531521679 150 -0.092823921 1.000 0.960 -0.09430729
rs2017143          1    2252811 150  0.023731274 1.000 1.000  0.05393011
rs2840531          1    2318570 150 -0.054122537 1.000 1.000 -0.05682457
rs2477703          1    2458154 150 -0.026077897 1.000 1.000 -0.03853168
rs734999           1    2545378 150 -0.001253863 1.000 1.000  0.06778755
rs2377041          1    2736485 150 -0.012215600 1.000 1.000  0.02296854
                 effBB P2df
rs1075456 -0.224121723  0.6
rs1054889  0.223671599  1.0
rs1781670  0.184368208  1.0
rs1264007 -0.151201249  1.0
rs774033  -0.185604344  1.0
rs2017143  0.037645098  1.0
rs2840531 -0.083781477  1.0
rs2477703 -0.038971061  1.0
rs734999  -0.006219629  1.0
rs2377041 -0.027814290  1.0
```

As you can see, now the "top" hit starts approaching genome-wide significance (genome-wide $P-value \sim 10\%$), showing the power of kinship-based methods under high heritability model.

Finally, let us plot $-log_{10}$ nominal $P-values$ from different methods across the genome. Let black dots correspond to GC, green to GRAMMAS and red to FASTA (figure 6.4):

```
> plot(mms, df = "Pc1df")
> add.plot(grs, df = "Pc1df", col = c("lightgreen", "lightblue"),
```

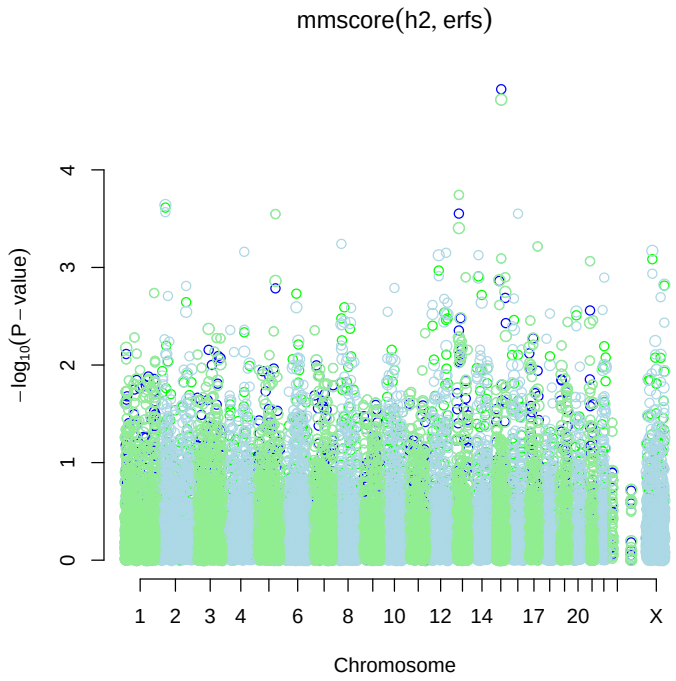

Figure 6.4: Comparison of FASTA (red), GRAMMAS (green), and plain GC (black).

```
+     cex = 1.2)
> add.plot(qts)
```

You can see that there is a great degree of correlation between the FASTA and GRAMMAS $P-values$, while plain GC really stands apart.

6.4 Exercise: analysis of family data

Exercise 30 *Repeat heritability estimation, FASTA and GRAMMAS analysis of previous section using pediree kinship (`pkins` object). Discuss the results.*

In the next section, you will explore a small (695 people) subset of people from ERF, a family-based study with participants coming from a genetically isolated population and sampled based on kinship (all living descendants of 22 couples living in the area in mid-XIXth century). The study participants were genotyped using Illumina 6K "linkage" array. QC was already performed. Your trait of interest is "qtbas".

Explore the data set and answer the questions:

Exercise 31 *Describe the trait "qt". Can you detect significant outliers at visual inspection? Is trait distributed normally? What are significant covariates?*

Exercise 32 *Explore relations between genomic and pedigree kinship (these are provided in data as* `gkin` *and* `pkin` *data objects, respectively). What are your conclusions? Which matrix would you use later on?*

Exercise 33 *What is the heritability of the trait (take care: polygenic analysis may rung for a long while)? Based on heritability analysis, how would you rank different methods of GWA analysis for this trait (and why)?*

Exercise 34 *Do GWA analysis using simple score test with genomic control. Estimate genome-wide significance. What are your conclusions?*

Run GWA analysis using the "best" method and model as you have decided in previous exercises. Estimate genome-wide significance. What are your conclusions? Did they change compared to simple analysis?

Exercise 35 *Repeat the last "best" analysis using pedigree kinship. How your results change?*

Exercise 36 *If you have any time left – repeat analysis using "qt" trait. This one is much more fun, but also more laborous to analyse.*

Chapter 7

Exploring and using public databases

See help for `show.ncbi`

More functions coming soon.

Chapter 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")
```

Chapter 9

Imperfect knowledge about genotypes

This section is copy-pasted from `ProbABEL` manual – will be re-written.

Many statistical and experimental techniques, such as imputations and high-throughput sequencing, generate the data, which are informative for genome-wide association analysis, and are probabilistic in the nature.

When we work with directly genotyped markers using such techniques as SNP or microsatellite typing, we would normally know the genotype of a particular person at a particular locus with very high degree of confidence, and, in case of biallelic marker, can state whether genotype is AA, AB or BB.

On the contract, when dealing with imputed or high-throughput sequencing data, for many of genomic loci we are quite uncertainty about genotypic status of the person; instead of known genotypes we deal rather with probability distribution; that is based on observed information, we have estimates that true underlying genotype is either AA, AB or BB; the degree of confidence about the real status is measured with probability distribution $\{P(AA), P(AB), P(BB)\}$.

Several techniques may be applied to analyse such data. The most simplistic approach would be to pick up the genotype with highest probability, i.e. $\max_g[P(g = AA), P(g = AB), P(g = BB)]$ and then analyse the data as if directly typed markers were used. The disadvantage of this approach is that it does not take into account the probability distribution – i.e. the uncertainty about the true genotypic status. Such analysis is statistically wrong: the estimates of association parameters (regression coefficients, odds or hazard ratios, etc.) are biased, and the bias becomes more pronounced with greater probability distribution uncertainty (entropy).

One of the solution which generates unbiased estimates of association parameters and takes probability distribution into account is achieved by performing association analysis by means of regression of the outcome of interest onto estimated genotypic probabilities.

9.1 Regression analysis

Standard linear theory is used to estimate betas and their standard errors. We assume linear model with expectation

$$E[\mathbf{Y}] = \mathbf{X}\beta \tag{9.1}$$

and variance-covaraince matrix

$$\mathbf{V} = \sigma^2\mathbf{I}$$

where $\mathbf{Y}$ is the vector of phenotypes of interest, $\mathbf{X}$ is design matrix, β is the vector of regression parameters, σ^2 is variance and $\mathbf{I}$ is identity matrix.

The maximum likelihood estimates (MLEs) for the regression parameters is given by

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{9.2}$$

and MLE of the variance is

$$\hat{\sigma}^2 = \frac{(\mathbf{Y} - \mathbf{X}\hat{\beta})^T(\mathbf{Y} - \mathbf{X}\hat{\beta})}{N - r_X} \tag{9.3}$$

where N is the number of observations and r_X is rank of $\mathbf{X}$ (number of columns of the design matrix).

Standard errors for the j-th parameter can be obtained as

$$s.e.(\hat{\beta}_j) = \hat{\sigma}^2(\mathbf{X}^T\mathbf{X})^{-1}_{jj} \tag{9.4}$$

where $(\mathbf{X}^T\mathbf{X})^{-1}_{jj}$ stands for the j-th diagonal element of the inverse of matrix $(\mathbf{X}^T\mathbf{X})$.

Logistic and survibal to be filled in here...

9.2 Analysis of imputed data with `ProbABEL`

`ProbABEL` takes three files as input: a file containing SNP information (e.g. MLINFO file of MACH), file with genome- or chromosome-wide predictor information (e.g. MLDOSE file of MACH), and a file containing phenotype of interest and covariates.

Optionally, the map information can be supplied (e.g. "legend" files of HapMap).

SNP information file

In the simplest scenario, SNP information file is an MLINFO file generated by MACH. This must be a space or tab-delimited file containing SNP name, coding for allele 1 and 2 (e.g. A, T, G or C), frequency of allele 1, minor allele frequency and two quality metrics ("Quality" = average maximum posterior probability and "Rsq" – ratio between observed and expected variances).

Actually, for `ProbABEL`, it does not matter what is written in this file – this information is just brought forward to the output. However, **it is critical** that the number of columns is seven and the number of lines in the file is equal to the number of SNPs in the corresponding DOSE file (plus one for the header line).

The example of SNP information file content follows here (also to be found in `ProbABEL/example/test.mlinfo`)

Note that header line is present in the file. The file describes five SNPs.

Genomic predictor file

Again under simplest scenario this is a MLDOSE file generated by MACH. Such file starts with two special columns plus, for each of the SNPs under consideration, a column containing the estimated allele 1 dose. The first "special" column is made of the sequential id, followed by an arrow followed by study ID (the one specified in MACH input files). The second column contains method (e.g. "MLDOSE") keyword.

An example of the few first lines of an MLDOSE file for five SNPs described in SNP information file follows here (also to be found in `ProbABEL/example/test.mldose`)

The order of SNPs in the SNP information file and DOSE-file must be the same. This should be the case if you just used MACH outputs.

Thus, by all means, the number of columns in the genomic predictor file must be the same as the number of lines in the SNP information file plus one.

Phenotypic file

Phenotypic data file contains phenotypic data, but also specifies the analysis model. There is a header line, specifying the variable names. The first column should contain personal study IDs. It is assumed that **both the total number and the order of these IDs is are exactly the same as in the genomic predictor (MLDOSE) file described in previous section**. This is not difficult to arrange using e.g. `R`; example is given in `ProbABEL/examples` directory.

Missing data should be coded with 'NA', 'N' or 'NaN' codes. Any other coding will be converted to some number which will be used in analysis! E.g. coding missing as '-999.9' will result in analysis which will consider -999.9 as indeed true measurements of the trait/covariates.

In case of linear or logistic regression (programs `palinear` and `palogist`, respectively), the second column specifies the trait under analysis, while the third, fourth, etc. provide information on covariates to be included into analysis. An example few lines of phenotypic information file designed for linear regression analysis follow here (also to be found in `ProbABEL/example/height.txt`)

Note again that the order of IDs is the same between MLDOSE and phenotypic data file. The model specified by this file is $height \sim \mu + sex + age$, where μ is intercept.

Clearly, you can for example include `sex x age` interaction terms by specifying another column having a product of sex and age here.

For logistic regression, it is assumed that in the second column cases are coded as "1" and controls as "0". An example few lines of phenotypic information file designed for logistic regression analysis follow here (also to be found in `ProbABEL/example/logist_data.txt`)

You can see that in the first 10 people, there are three cases, as indicated by "chd" equal to one. The model specified by this file is $chd \sim \mu + sex + age + othercov$.

In case of Cox proportional hazards model, the composition of the phenotypic input file is a bit different. In the second column and third column, you need to specify the outcome in terms of follow-up time (column two) and event (column three, "1" if event occurred and zero if censoring). Columns from four inclusive specify covariates to be included into analysis. An example few lines of

phenotypic information file designed for Cox proportional hazards model analysis follow here (also to be found in `ProbABEL/example/coxph_data.txt`)

You can see that for first 10 people, event happens for three of them, while for the other seven there is no event during follow-up time, as indicated by "chd" column. Follow-up time is specified in the preceding column. The covariates included into the model are age (presumably at baseline), sex and "othercov"; thus the model, in terms of `R/survival` is $Surv(fuptime\_chd, chd) \sim sex + age + othercov$.

Optional map file

If you would like that map information (e.g. base pair position) to be included in your outputs, you can supply a map file. These follow HapMap "legend" file format. For example, for the five SNPs we considered the map-file may look like

The order of the SNPs in the map file should follow that in the SNP information file. Only information from the second column – the SNP location – is actually used to generate the output.

Running analysis

To run linear regression, you should use program called `palinear`; for logistic analysis use `palogist`, and for Cox proportional hazards model use `pacoxph` (to be found in `ProbABEL/bin/` directory after you have compiled the program).

There are in total 11 command line options you can specify to `ProbABEL` analysis functions `linear` or `logistic`. If you run either program without any argument, you will get a short explanation to command line options:

```
user@server~$ palogist

Usage: ../bin/palogist options
Options:
         --pheno   : phenotype file name
         --info    : information (e.g. MLINFO) file name
         --dose    : predictor (e.g. MLDOSE) file name
         --map     : [optional] map file name
         --nids    : [optional] number of people to analyse
         --chrom   : [optional] chromosome (to be passed to
                                                    output)
         --out     : [optional] output file name (default is
                                        regression.out.csv)
         --skipd   : [optional] how many columns to skip in
                              predictor (dose) file (default 2)
         --ntraits : [optional] how many traits are analysed
                                                  (default 1)
         --separat : [optional] character to separate fields
                                         (default is space)
         --no-head : do not report header line
         --help    : print help
```

However, for a simple run you can use only three, which specify the necessary files needed to run regression analysis.

These options are `-dose` (or `-d`), specifying genomic predictor / MLDOSE file described in sub-section 9.2; `-pheno` (or `-p`), specifying the phenotypic data

file described in sub-section 9.2; and `-info` (or `-i`), specifying the SNP information file described in sub-section 9.2.

If you change to the `ProbABEL/example` directory you can run analysis of height by

```
user@server~/ProbABEL/example/$ ../bin/palinear -p height.txt
                                -d test.mldose -i test.mlinfo
```

Output from analysis will be directed to "regression.out.csv" file.

You can run analysis of binary trait "chd" with

```
user@server~/ProbABEL/example/$ ../bin/palogist -p logist_data.txt
                                -d test.mldose -i test.mlinfo
```

To run a Cox proportional hazards model, try

```
user@server~/ProbABEL/example/$ ../bin/pacoxph -p coxph_data.txt
                                -d test.mldose -i test.mlinfo
```

Please have a look at the shell script files `example_qt.sh`, `example_bt.sh` and `example_all.sh` to have a better overview of analysis options.

Output file format

Let us consider what comes out of the linear regression analysis described in the above section. After you have run the analysis, in the output file you will find something like

Here, I show only three first lines of output. Note that lines starting with "..." are actually the ones continuing the previous line – I just have wrapped this output so we can see these long lines.

The header provides short description of what can be found in the specific column. The first column provides the SNP name and next six are descriptives which were brought directly from the SNP information file. Thus these describe allele frequencies and quality in your total imputations, not necessarily in the data under analysis.

On the contrast, starting with the next column, named "n", the output concerns the data analysed. Column 8 ("n") tells the number of subjects for whom complete phenotypic information was available. At this point, unless you have complete measurements on all subjects, you should feel warned if the number here is exactly the number of people in the file – this probably indicates you did not code missing values according to `ProbABEL` format ('NA', 'NaN', or 'N').

The next column nine ("Mean_predictor/2") gives you the mean value of the predictor (SNP) divided by 2. When MLDOSE file is used, this corresponds to the estimated frequency of allele 1 in people with complete phenotypic data.

If "–chrom" option was used, in the next column you will find the value specified by this option. If "–map" option was used, in next column you will find map location brought from the map-file. Next columns, starting with "beta_mu", provide estimates for the regression coefficients ("beta_mu" is a bit awkward name for intercept). The number of βs reported is equal to the number of covariates in the phenotypic file plus two (intercept and β_{SNP}). It is likely that you will be most interested in "beta_SNP" – the estimate of the regression coefficient for the SNP in consideration. Next columns provide corresponding

standard errors of the estimates ("sebeta_mu", ... "sebeta_SNP"). Column names "sigma2" provides residual variance; finally, "SNP_Z" provides the value of the Wald's test computed as

$$Z_{SNP} = \frac{\beta_{SNP}}{s.e._{\beta_{SNP}}}$$

while "SNP_chi2" provides the χ^2_1 test obtained as square of Z_{SNP}.

Preparing input files

In the `ProbABEL/bin` directory you can find `perepare_data.R` file – an R script which arranges phenotypic data in right format. Please read this script for details.

Memory use and performance

Maximum likelihood regression is implemented in `ProbABEL`. With 6,000 people and 2.5 millions SNPs, genome-wide scan is completed in less that an hour for linear model with 1-2 covariates and overnight for logistic regression or Cox proportional hazards model.

Memory is an issue with `ProbABEL` – large chromosomes, such as chromosme one consumed up to 5Gb RAM with 6,000 people.

Chapter 10

Meta-analysis of GWA scans

10.1 Standard meta-analysis methods

Imagine you are interested in the effect of a certain polymorphism onto a particular disease. After scanning literature, you find some studies that implicate certain allele as significantly increasing the risk of the disease, but you will typically find also that other studies were inconclusive (no significant association), and that even some of the studies implicated the same allele as "protective". Your gut feeling may be that the allele is indeed the risk one, because you feel that the studies contradicting to this hypothesis were based on small number of subjects; however, how do you quantify this feeling? In this situation you need to perform meta-analysis of available data to come up with the joint effect size estimate and P-value, as based on all available data.

Let us first consider a situation when you are interested in the effect of the allele on a quantitative phenotype, expressed as a coefficient of regression of the trait onto the number of this allele in the genotype. Under a favourable scenario, from every individual study you would know the estimate of this regression coefficient, and the standard error of the estimate (or, equivalently, the $P-value$ or the test statistics value for association).

One of approaches frequently used in meta-analysis of the data coming from a number of independent studies is the inverse variance method. In essence, this method is equivalent to combining likelihoods coming from separate studies, using quadratic approximation. Denote coefficients of regression estimated in N studies as β_i, and associated squared standard errors of the estimates as s_i^2 where $i \in 1, 2, ..., N$. Note that the regression coefficient should be reported on the same scale, e.g. centimeters, meters, or using observations reported on the standard normal scale. Define weights for individual studies as

$$w_i = \frac{1}{s_i^2}$$

Then the pooled estimate of the regression coefficient is

$$\beta = \frac{\sum_{i=1}^{N} w_i \beta_i}{\sum_{i=1}^{N} w_i}$$

As you can see, the weights have straightforward interpretation: the bigger the weight of the study (meaning the small is the standard error in the study), the larger is the contribution from this study onto the pooled estimate.

The standard error of the pooled estimate is computed as

$$s^2 = \frac{1}{\sum_{i=1}^{N} w_i}$$

and the χ^2-test for association is computed in standard manner as

$$T^2 = \frac{\beta^2}{s^2} = \frac{\left(\sum_{i=1}^{N} w_i \beta_i\right)^2}{\sum_{i=1}^{N} w_i}$$

or, alternatively, the Z-test is

$$Z = \frac{\beta}{s} = \frac{\sum_{i=1}^{N} w_i \beta_i}{\sqrt{\sum_{i=1}^{N} w_i}}$$

Let us try to do meta-analysis using the inverse variance pooling method. Imagine we have information from four different studies reporting effect and the standard error of the same allele:

Table 10.1: Estimated regression coefficients from four studies

Study	n	β	s_β	χ^2
1	225	0.16	0.07	5.224
2	560	0.091	0.042	4.694
3	437	0.072	0.048	2.25
4	89	-0.03	0.12	0.062
Total	1311	?	?	?

Let us try to access the joint significance of the association using these data. First, let us define a vector of regression coefficients and squared standard errors:

```
> beta <- c(0.16, 0.091, 0.072, -0.03)
> s <- c(0.07, 0.042, 0.048, 0.12)
> s2 <- s * s
> s2
```

```
[1] 0.004900 0.001764 0.002304 0.014400
```

Compute the weight for individual studies as

```
> w <- 1/s2
> w
```

```
[1] 204.08163 566.89342 434.02778  69.44444
```

Estimate pooled regression coefficient as

```
> pbeta <- sum(w * beta)/sum(w)
> pbeta
```

```
[1] 0.08898527
```

and pooled squared standard error as

```
> ps2 <- 1/sum(w)
> ps2
```

```
[1] 0.0007846539
```

To access significance of association in meta-analysis, let us compute χ^2 test statistics and the $P - value$ with

```
> pchi2 <- pbeta * pbeta/ps2
> pchi2
```

```
[1] 10.09155
```

```
> ppvalue <- 1 - pchisq(pchi2, 1)
> ppvalue
```

```
[1] 0.001489504
```

We conclude that there is a significant association in meta-analysis.

There is an important effect which should be considered when doing meta-analysis of published data. Given numerous polymorphisms available in human genome, a particular polymorphism usually becomes a focus of interest only when it was shown to be significantly associated in some study which reports it. Put it other way around: only when a significant association was detected and reported, more studies are likely to be performed on the same polymorphism. This first report, however, is not guaranteed to demonstrate a true association: it may well report a false-positive, or, even in presence of association, it is likely to over-estimate the effect of the polymorphism. Thus there is a positive bias in literature reports, and this bias is particularly strong for the first report, a phenomenon frequently referenced to as "champion's" or "winner's curse".

The observations we have just considered are quite typical in that the first study, where the association was originally discovered, reports the biggest effect and most significant effect, while the follow-up studies suggest smaller effect.

Therefore, when you meta-analyse data from publications it is always good idea to exclude the first report (in case it is positive – and it is always positive!) and check if significant association is still observed. Let us try to do that:

```
> beta <- beta[2:4]
> s2 <- s2[2:4]
> w <- w[2:4]
> pbeta <- sum(w * beta)/sum(w)
> pbeta
```

```
[1] 0.07544522
```

```
> ps2 <- 1/sum(w)
> ps2
```

```
[1] 0.0009342602
```

```
> pchi2 <- pbeta * pbeta/ps2
> pchi2

[1] 6.092501

> ppvalue <- 1 - pchisq(pchi2, 1)
> ppvalue

[1] 0.01357568
```

Indeed, when the first "champion" report is excluded, the overall evidence is decreased and results become less significant, though still pointing to the same direction.

When binary traits are studied, and results are reported as Odds Ratios with $P - values$, it is also possible to apply inverse variance method. For this, you need to transform your Odds Ratios using natural logarithm, and, on this scale, estimate the standard error. Generic inverse variance pooling may be applied to the data transformed this way; the final results are back-transformed onto Odds Ratio scale using exponentiation.

Let us consider a simple example. Let Odds Ratios and χ^2 test statistics values coming from four studies of a binary phenotype are as following: $\theta_1 = 1.5$ ($\chi^2 = 5.1$), $\theta_2 = 1.3$ ($\chi^2 = 2.2$), $\theta_3 = 0.9$ ($\chi^2 = 0.5$), $\theta_4 = 1.2$ ($\chi^2 = 3.1$).

Let us first transform the Odds Ratio to the logarithmic scale with

```
> or <- c(1.5, 1.3, 0.9, 1.2)
> lnor <- log(or)
> lnor

[1]  0.4054651  0.2623643 -0.1053605  0.1823216
```

To compute standard errors from known χ^2 values, one can use simple relation

$$\chi^2 = \frac{\beta^2}{s^2}$$

and thus

$$s^2 = \frac{\beta^2}{\chi^2}$$

Thus to compute the square standard errors corresponding to the log-Odds Ratio, we can use

```
> chi2or <- c(5.1, 2.2, 0.5, 3.1)
> s2lnor <- lnor * lnor/chi2or
> s2lnor

[1] 0.03223568 0.03128864 0.02220168 0.01072295
```

Now we can combine log-Odds Ratios and corresponding standard errors using inverse variance method:

```
> w <- 1/s2lnor
> plnor <- sum(w * lnor)/sum(w)
> plnor
```

```
[1] 0.1650462
```

```
> ps2 <- 1/sum(w)
> ps2
```

```
[1] 0.004968165
```

```
> pchi2 <- plnor * plnor/ps2
> pchi2
```

```
[1] 5.482958
```

```
> ppval <- 1 - pchisq(pchi2, 1)
> ppval
```

```
[1] 0.01920274
```

And the corresponding estimate of pooled Odds Ratio is

```
> exp(plnor)
```

```
[1] 1.179448
```

and 95% confidence interval is

```
> exp(plnor - 1.96 * sqrt(ps2))
```

```
[1] 1.02726
```

```
> exp(plnor + 1.96 * sqrt(ps2))
```

```
[1] 1.354181
```

Some times, effects are reported on different scales, and/or there may be suspect that these effects are not translatable across studies because of the differences in experimental design or for some other reasons. In this case, it may be better to poll the data without use of the effect estimate in exact manner, based only on the sign of association and its significance. This can be done by pooling Z-score values. Z-score refers to the test statistics, which has standard normal distribution under the null and can be derived e.g. by dividing estimate of the regression coefficient onto its standard error:

$$Z_i = \frac{\beta_i}{s_i}$$

The Z-score pooling methods can be derived from the inverse variance pooling by exploiting the fact that generally standard error of the estimate is proportional to $1/\sqrt{n}$, where n is the number of observations used for estimation. Therefore individual scores are assigned weights which are proportional to the square root of number of independent observations used in individual study, $w_i = \sqrt{n_i}$. The pooled Z-score statistics is computed as

$$Z = \frac{\sum_{i=0}^{N} w_i Z_i}{\sqrt{\sum_{i=0}^{N} w_i^2}}$$

We can now repeat the analysis of our first data set using Z-score pooling method. First, our data from table 10.1 are

```
> n <- c(225, 560, 437, 89)
> beta <- c(0.16, 0.091, 0.072, -0.03)
> s <- c(0.07, 0.042, 0.048, 0.12)
```

The Z-scores and weights are are:

```
> z <- beta/s
> z

[1]  2.285714  2.166667  1.500000 -0.250000

> w <- sqrt(n)
> w

[1] 15.000000 23.664319 20.904545  9.433981
```

The pooled estimate of Z-score is

```
> pz <- sum(w * z)/sqrt(sum(w^2))
> pz

[1] 3.163875
```

and corresponding $P - value$ is

```
> 1 - pchisq(pz * pz, 1)

[1] 0.001556839
```

which is almost the same $P - value$ we have obtained previously using the inverse variance method. Note, however, that now we do not know the "pooled" estimate of the regression coefficient.

Other important aspects of meta-analysis, such as heterogeneity, and a wide range of methods different from the inverse variance and Z-score based methods are not covered here, and we refer the reader to more epidemiologically-oriented literature for a better review.

10.2 Exercise: meta-analysis of literature data

In this exercise, you will perform meta-analysis of results collected from literature. These results resemble these obtained for association analysis between Pro12Ala polymorphism of the PPAR-γ gene and type 2 diabetes. The data collected from literature are presented in the table 10.2.

As you can see, only the original study report significant association, while other four are insignificant and one point in opposite direction.

Answer the following questions:

Exercise 37 *Perform meta-analysis of the data presented in table 10.2. Which allele is the risk one? Is this risk significant? What is pooled Odds Ratio and 95% confidence interval? Do analysis using at least two methods. Which method is better (best) in this situation? Why?*

Exercise 38 *Perform meta-analsys excluding the original report (study 1). Is there still significant association between Pro12Ala and diabetes?*

Table 10.2: Summary of six studies of association between T2D and Pro12Ala polymorphism of the PPAR-γ gene. n: number of subjects; effective allele: the allele for which the OR was estimated.

Study	Effective allele	n	OR_E	$P-value$
1	Ala	221	0.67	0.013
2	Pro	306	0.93	0.60
3	Pro	71	1.08	0.84
4	Ala	164	0.83	0.40
5	Pro	242	1.22	0.25
6	Pro	471	1.23	0.07

10.3 Reporting GWA results for future meta-analysis

In this section, we will discuss specifics of GWA analysis when meta-analysis is aimed at later stage. In order to perform meta-analysis at later stage, using either inverse variance or Z-score based method, you generally need to report only effect estimates, standard errors of The estimates (or, equivalently, $P-values$ or test statistics values), and number of observations used for estimation.

It is also clear that it is crucial to know for which allele the effect is reported, and this is the point where meta-analysis of genetic data may be very confusing. Generally, one may think that reporting what couple of nucleotide bases correspond to the polymorphism under the study and defining what allele was used as reference in the regression model may be enough. This, however, is not true for certain class of polymorphisms and may be a source of great confusion.

Consider a DNA molecule; as you know it is made of two complementary strands (forward or "+" and reverse or "-"). As you may guess, depending on the strand, what is an "A/G" polymorphism when reported on "+" strand becomes "T/C" polymorphism on the "-" strand (using complementarity rule A<->T and G<->C). This is not a big problem for most of the polymorphism classes, because if say you know that for a first study β_1 is reported for the "G" allele of the "A/G" polymorphism and in the second study β_2 is the estimate of the effect "C" allele of the same polymorphism, but coded as "T/C" (thus other strand), you can easily spot that and say the "C" is the same as "G" in this situation, and pool the two betas straightforwardly.

However, for two types of polymorphisms, "A/T" and "G/C", where you can not get away without knowing what the strand was: what is reported as the effect of "T" in "A/T" polymorphism in study one; and seemingly corresponding effect of "T" in "A/T" polymorphism in study two may be apparently reports for two opposite alleles, if strands used for reporting in two studies were different.

The story may become even more complex, because the forward/reverse orientation depends on the genomic build[1].

Thus if you want to pool your results with the results of others, there are quite a few SNP characteristics which are absolutely crucial to report, namely, the nucleotide bases describing the polymorphism, with indication which one was used as reference, and which one as "effective", strand, genomic build, and only

[1] There is alternative, top/bottom, strand designation, which does not depend on genomic build. However, it is not always used.

than the effect estimate, standard error of the effect, and number of observations used to do estimation.

Other characteristics which are also recommended for reporting because they describe the quality characteristics of the sample and/or provide redundant information, which is good for double checks. Such characteristics include: frequency of the effective or reference allele, call rate, P-value for Hardy-Weinberg equlibrium and may be some parameter describing what is the direction of deviation from HWE (e.g. F_{max}). When reporting results for imputed SNPs, more quality control characteristics should be included, suh as average maximal posterior probability, R^2, etc.

Let us start with arranging two data sets which could then be used for meta-analysis. Basically, we will use cleaned data from the GWA exercise you did in section 5 (5, page 67), and split that is two parts.

Let us first load the data and re-name the data object:

```
> data(ge03d2ex.clean)
> data2 <- ge03d2ex.clean
> rm(ge03d2.clean)
```

and then split it in two parts:

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

```
[1] 125
```

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

We will analyse body mass index. If you pooling results of analysis of studies which are designed in approximately the same manner, you may think of reporting the effect estimates on the same scale and use of the inverse variance method for meta-analysis.

However, in meta-analysis of multiple data sets different individual studies are likely to assess different population, will use different designs, measure different covariates, and so on. Therefore you should think of some standardisation of the outcome variable (or apply Z-score method).

Therefore for the purpose of future meta-analysis, it becomes conventional to analyse pre-adjusted data which are scaled to Standard Normal (mean of zero and variance of unity). Note that this argument applies only to meta-analysis – you may and should report effects on the original scale (e.g. in centimeters and grams) in analysis of individual studies, in order to have better interpretability.

Moreover, in meta-analysis you heavily rely on the large numbers approximation when estimating $P-values$; while for individual study you can always apply empirical, e.g. permutation-based, procedures to derive the correct $P - value$ whatever is the distribution of the trait, in meta-analysis the Normality assumption becomes crucial, and you do not want few outliers spoiling your results by screwing up $P - values$. Therefore some transformation improving normality is desirable. Note that transformation to Standard Normal does not improve the fit to normality; to do that other transformation should be applied. Probably the most famous transformations are log- and square root ones, then one may think of Box-Cox transformation. At the same time there is a transformation,

called a Rank Transformation ro Normality which guarantees perfect fit to Normal in absence of heavy ties[2]. We advocate the use of Rank Transformation to Normal for meta-analysis purposes.

`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:

```
> zbmi0 <- ztransform(bmi, mdta1)
```

The histogram of the transformed variable and scatter-plot of raw against transformed BMI is given at figure 10.1, column 1. Note that the fit to Normality is not improved by this transformation; with the original BMI, the Shapiro test for deviation from normality gives

```
> shapiro.test(mdta1@phdata$bmi)

        Shapiro-Wilk normality test

data:  mdta1@phdata$bmi
W = 0.9328, p-value = 0.01990
```

with identical results from the transformed variable:

```
> shapiro.test(zbmi0)

        Shapiro-Wilk normality test

data:  zbmi0
W = 0.9328, p-value = 0.01990
```

This is quite natural: as you can note from scatter-plot in column 1 of figure 10.1, only the centering and the spread of the scales are different for X (original BMI) and Y (x0), otherwise there is an exact linear correspondence between the two.

We can also do transformation using sex and age-adjusted residuals with

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

The scatter-plot of raw against transformed BMI is given at figure 10.1, column 2. Note that this transformation may slightly change the fit to Normal, which happens because we factor out the effects of sex and age:

```
> shapiro.test(zbmi1)

        Shapiro-Wilk normality test

data:  zbmi1
W = 0.9263, p-value = 0.01224
```

[2] Ties are generated by the subjects with exactly the same trait values

From the scatter-plot in column 2 of figure 10.1, it is quite clear what happens: the residuals from linear regression are not corresponding to the original BMI in exact linear manner.

A similar function, which performs rank-transformation to normality, is named `rntransform`. For example if we want to adjust BMI for sex and age and rank-transform the residuals to Normal, we can use

```
> rnbmi1 <- rntransform(bmi ~ sex + age, mdta1)
```

This transformation, however, indeed improves the fit to Normal:

```
> shapiro.test(rnbmi1)

        Shapiro-Wilk normality test

data:  rnbmi1
W = 0.999, p-value = 1
```

In essence, the $P - value$ of 1 means perfect fit to Normal – and this is what should have occurred when this transformation is used on the data without ties. Perfectly Normal distribution of the transformed trait may be enjoyed at the histogram presented at column 3 of figure 10.1.

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

```
> qts1 <- qtscore(rnbmi1, 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 start yet: now we need to extract coding, reference allele, strand, etc. – otherwise we can not do right meta-analysis later on!

Let us assume that you want to summarise the GW results from additive 1 d.f. test using following variables: (1) SNP name (2) chromosome (3) position (4) number of people with data available for this SNP test (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).

In the above 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,

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

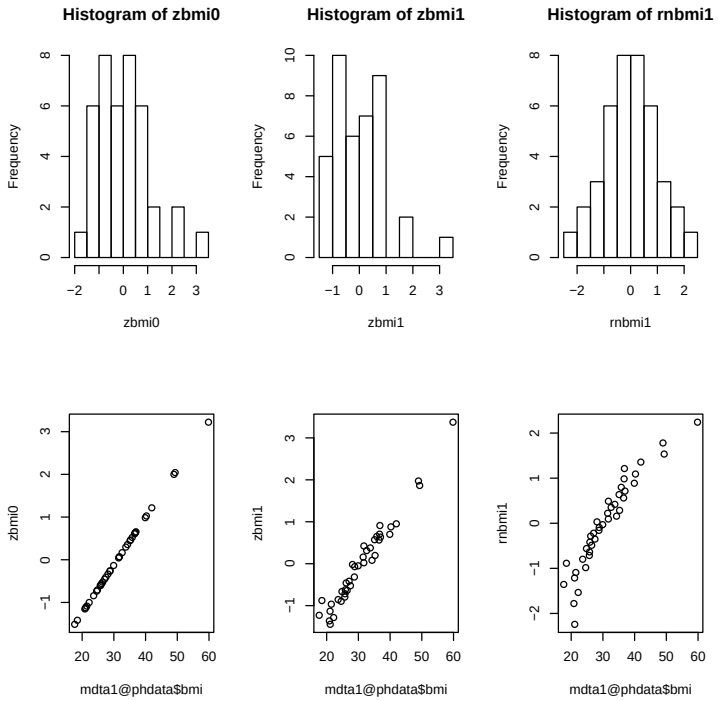

Figure 10.1: Histogram of transformed BMI and scatter-plots of the raw BMI against transformed BMI. Column 1: Z-transformation without covariates. Column 2: Z-transformation with adjustment for age and sex. Column 3: Rank-transformation to normality, after adjustment for sex and age.

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$effB, sebeta = seeff,
+     p = qts1$P1df, pgc = qts1$Pc1df, coding = as.character(mdta1@gtdata@coding),
+     strand = as.character(mdta1@gtdata@strand), reffreq = refallfreq)
```

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

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

```
              name chrom     pos  n       beta    sebeta         p       pgc
rs7435137 rs7435137     1 4259040 39 0.05838361 0.2141932 0.7851803 0.7851803
rs664063   rs664063     2 7288020 40 0.01829962 0.4705679 0.9689794 0.9689794
```

```
rs546570   rs546570     2 6120257 40 0.27341721 0.3890643 0.4822079 0.4822079
rs7908680 rs7908680     1 2311762 40 0.67913083 0.5908536 0.2503885 0.2503885
rs166732   rs166732     1 4716343 39 0.35418295 0.7213534 0.6234280 0.6234280
          coding strand   reffreq
rs7435137     CT      - 0.5128205
rs664063      GC      - 0.9375000
rs546570      TA      + 0.1000000
rs7908680     CA      - 0.9625000
rs166732      TG      - 0.9743590
```

However, it is not recommended that you perform above-described reporting actions unless you develop your own format. In case if you plan to use `MetABEL` for meta-analysis, you best use `formetascore` function, which basically performs operations similar to described, and reports results in format compatible with `MetABEL`.

Thus, if you plan to use `MetABEL` for meta-analysis, required tables can be generated with single command:

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

You can see that results are 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    0.48717949 39 0.05838361 0.2141932 0.7851803 0.7851803 0.7484124
rs664063     0.06250000 40 0.01829962 0.4705679 0.9689794 0.9689794 1.0000000
rs546570     0.90000000 40 0.27341721 0.3890643 0.4822079 0.4822079 1.0000000
rs7908680    0.03750000 40 0.67913083 0.5908536 0.2503885 0.2503885 1.0000000
rs166732     0.02564103 39 0.35418295 0.7213534 0.6234280 0.6234280 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.csv` function:

```
> write.csv(mdf1, file = "RData/part1.rnbmisexage.csv", row.names = F)
```

Similar analysis is applied to the second data set:

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

We can inspect the first five 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    0.47619048 84  0.1701252 0.1658456 0.3049836 0.3200269 0.2729091
rs664063     0.11309524 84  0.2235208 0.2428365 0.3573334 0.3722425 1.0000000
rs546570     0.95180723 83 -0.2957072 0.3683184 0.4220571 0.4364039 1.0000000
rs7908680    0.01234568 81 -0.2475810 0.7150117 0.7291462 0.7371260 1.0000000
rs166732     0.02976190 84 -0.3871952 0.4576614 0.3975362 0.4121416 1.0000000
              call
rs7435137 1.0000000
rs664063  1.0000000
rs546570  0.9880952
rs7908680 0.9642857
rs166732  1.0000000
```

Let us write the data to a file:

```
> write.csv(mdf2, file = "RData/part2.rnbmisexage.csv", row.names = F)
```

Finally let us analyse and save results for another data set, `ge03d2c`:

```
> data(ge03d2c)
> mdf3 <- formetascore(bmi ~ sex + age, ge03d2c, transform = rntransform)
> write.csv(mdf3, file = "RData/part3.rnbmisexage.csv", row.names = F)
```

10.4 Meta-analysis with MetABEL

Now we are ready to meta-analyse GWA data coming from three studies. For this we will need to use `MetABEL` package, implementing simple meta-analysis functions for GWA data. Start with loading the package:

```
> library(MetABEL)
```

We will first meta-analyse the three studies using the data frames generated in previous section, `mdf1`, `mdf2` and `mdf3`. For this we will use the core function of `MetABEL`, `metagwa.tables`. This function takes four arguments: two data frames with results from individual studies, and two arguments supplying the study names. Pooling of multiple studies is possible by sequential application of this function.

Let us pool two first data frames:

```
> pooled <- metagwa.tables(mdf1, mdf2, name.x = "Part1", name.y = "Part2")
```

```
analysing ...
Lambda Part1 = 0.9440436
Lambda Part2 = 1.090242
Corrected Lambda Part1 = 0.9440436
Corrected Lambda Part2 = 1
Lambda POOLED data = 0.9997559
... DONE
```

The `pooled` data frame contains results of meta-analysis and essential details of the original studies:

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

```
       name strand allele1 allele2 effallele chromosome position   n npops
1  rs100616      -       G       C         C          1  1911712 123     2
2 rs1006497      +       T       G         G          1  2658810 123     2
3 rs1011580      +       A       G         G          3 10048771 122     2
4 rs1011953      +       A       G         G          2  6464510 124     2
5 rs1013473      +       A       T         T          1  4487262 124     2
         beta    sebeta effallelefreq      call    pexhwe  obetaPart1
1  0.11128490 0.1842895     0.1260163 0.9919667 4.7183287  0.08819818
2 -0.07530695 0.1933475     0.1463415 0.9920732 0.1698876 -0.25425751
3 -0.06827202 0.1394457     0.5040984 0.9839091 3.8552114 -0.06755996
4  0.07615424 0.1367472     0.3306452 1.0000000 0.1037354  0.18982068
5  0.30191078 0.1318863     0.5161290 1.0000000 0.0000000  0.31522388
    obetaPart2  osePart1  osePart2      chi2          p
1  0.118757384 0.3726827 0.2120267 0.3646457 0.54593737
2  0.003368901 0.3498751 0.2319889 0.1517026 0.69691357
3 -0.068805418 0.2130800 0.1844217 0.2397041 0.62441989
4  0.007546243 0.2228918 0.1731671 0.3101352 0.57759723
5  0.295031422 0.2259542 0.1624257 5.2403198 0.02206922
```

If one needs to pool more studies, this data frame should be used as the first argument of the `metagwa.tables`, and `name.x` argument should take special value "POOLED":

```
> pooled <- metagwa.tables(pooled, mdf3, name.x = "POOLED", name.y = "mdf3")
```

```
NA for betas in both populaions
18 SNPs removed
analysing ...
Lambda mdf3 = 1.128096
Corrected Lambda mdf3 = 1
Lambda POOLED data = 1.357809
... DONE
```

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

```
       name strand allele1 allele2 effallele chromosome position   n npops
1 rs1000475      +       T       C         C          X 13721802  91     1
2 rs1000909      -       A       G         G          2  8531681 190     1
3 rs1006092      -       T       G         G          X 13527448 190     1
4  rs100616      -       G       C         C          1  1911712 311     3
5 rs1006497      +       T       G         G          1  2658810 316     3
         beta     sebeta effallelefreq      call    pexhwe  obetaPart1
1 -0.05521349 0.11185255     0.5824176 0.4690722 5.9671619          NA
2 -0.08504871 0.13218866     0.8157895 0.9793814 2.8521518          NA
3 -0.03712065 0.08523891     0.5078947 0.9793814 6.8378385          NA
4  0.11688313 0.12065072     0.1237942 0.9781269 6.6661003  0.08819818
5 -0.02265030 0.11396275     0.1613924 0.9937663 0.2330977 -0.25425751
```

```
   obetaPart2    obetamdf3  osePart1  osePart2    osemdf3       chi2         p
1          NA -0.055213494        NA        NA 0.11185255 0.24366808 0.6215693
2          NA -0.085048708        NA        NA 0.13218866 0.41394922 0.5199718
3          NA -0.037120652        NA        NA 0.08523891 0.18965112 0.6632071
4 0.118757384  0.121082405 0.3726827 0.2120267 0.15961076 0.93852071 0.3326586
5 0.003368901  0.005382408 0.3498751 0.2319889 0.14107317 0.03950226 0.8424569
```

This procedure may become quite laborious if multiple studies are to be pooled. In this case, it is possible to run meta-analysis using data provided in files, by applying function `metagwa.files`. As the first argument, this function takes the path to the directory where the files with results of individual studies are stored. It is assumed that the file names are made of two parts: population/study name and an extension. Thus the second argument of the `metagwa.files` function is the vector with names of studies, and the third one provides extension. Other arguments, "maf", "call" and "phwe" provide the threshold for QC filtering of SNPs in individual studies.

The function does not return any value, but rather creates a new file named `POOLEDextens`, where "extens" is the argument supplied to the function, in the source directory. To run analysis on the three files in the directory "RData" we can use

```
> metagwa.files(dir = "RData", pops = c("part1", "part2", "part3"),
+     extens = ".rnbmisexage.csv", maf = 1e-04, call = 0.93, phwe = 1e-08)

Population part1 , reading RData/part1.rnbmisexage.csv done
Dimesions after filters are 3678 15
population part2, reading RData/part2.rnbmisexage.csv done
Dimesions after filters are 3823 15
analysing ...
Lambda part1 = 0.9403026
Lambda part2 = 1.090242
Corrected Lambda part1 = 0.9403026
Corrected Lambda part2 = 1
Lambda POOLED data = 0.9971353
... DONE
Dimesions after pooling are 3868 20
population part3, reading RData/part3.rnbmisexage.csv done
Dimesions after filters are 7493 15
analysing ...
Lambda part3 = 1.127385
Corrected Lambda part3 = 1
Lambda POOLED data = 1.360099
... DONE
Dimesions after pooling are 7493 22
$analysed.pops
[1] "part1" "part2" "part3"
```

```
> poolf <- read.csv("RData/POOLED.rnbmisexage.csv", strings = F)
> poolf[1:5, ]

       name strand allele1 allele2 effallele chromosome position   n npops
1 rs1000909      -       A       G         G          2  8531681 190     1
```

```
2 rs1006092      -       T       G         G          X 13527448 190     1
3  rs100616      -       G       C         C          1  1911712 311     3
4 rs1006497      +       T       G         G          1  2658810 316     3
5 rs1010481      +       A       C         C          2  8409087 190     1
         beta     sebeta effallelefreq      call    pexhwe  obetapart1
1 -0.08504871 0.13214698     0.8157895 0.9793814 2.8521518          NA
2 -0.03712065 0.08521203     0.5078947 0.9793814 6.8378385          NA
3  0.11688465 0.12062898     0.1237942 0.9781269 6.6661003  0.08819818
4 -0.02263876 0.11393929     0.1613924 0.9937663 0.2330977 -0.25425751
5 -0.03241475 0.12373958     0.2657895 0.9793814 0.0000000          NA
   obetapart2   obetapart3  osepart1  osepart2   osepart3       chi2         p
1          NA -0.085048708        NA        NA 0.13214698 0.41421039 0.5198402
2          NA -0.037120652        NA        NA 0.08521203 0.18977078 0.6631075
3 0.118757384  0.121082405 0.3726827 0.2120267 0.15956043 0.93888336 0.3325652
4 0.003368901  0.005382408 0.3498751 0.2319889 0.14102869 0.03947828 0.8425041
5          NA -0.032414750        NA        NA 0.12373958 0.06862274 0.7933527
```

Chapter 11

Analysis of selected region

11.1 Exploring linkage disequilibrium

See help for `r2fast`.

11.2 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))
```

11.3 Analysis of interactions

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

Appendix 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).

If you need to add pehnotypes to a `gwaa.data-class` object already created, you can use function `add.phdata`. This function allows you to add variables contained in some data frame to the existing `data@phdata` object. The data

frame to be added should contain "id" variable, identical to that existing in the object, and **should not** contain any othe variables with names identical to these already existing.

The second file you need should contains genotypic data. As described in section 4.1 (4.1, page 51), 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 rs123 rs6679 rs876 mitoA1
cd289982  "A/A"  "G/T"  "A/A"  "T/T" "G/T"  NA    "A/A"
cd325285  "A/G"  "G/G"  "C/C"  "T/G" "G/G"  NA    "C/C"
cd357273  "A/G"  "G/G"  "C/C"  "T/T" "G/G"  "T/T" NA
cd872422  "A/A"  "G/T"  "C/C"  "T/T" "G/T"  "G/G" NA
cd1005389 "G/G"  "G/G"  "C/A"  "T/T" "G/G"  "T/T" NA

> as.character(df@gtdata@strand)

rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
   "+"    "+"    "+"    "-"    "-"    "+"    "-"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
  "AG"   "GT"   "CA"   "TG"   "GT"   "TG"   "CA"
```

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 rs123 rs6679 rs876 mitoA1
cd289982  "A/A"  "G/T"  "A/A"  "T/T" "G/T"  NA    "A/A"
cd325285  "A/G"  "G/G"  "C/C"  "T/G" "G/G"  NA    "C/C"
cd357273  "A/G"  "G/G"  "C/C"  "T/T" "G/G"  "T/T" NA
cd872422  "A/A"  "G/T"  "C/C"  "T/T" "G/T"  "G/G" NA
cd1005389 "G/G"  "G/G"  "C/A"  "T/T" "G/G"  "T/T" NA

> as.character(df@gtdata@strand)

rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
   "+"    "+"    "+"    "+"    "+"    "+"    "+"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
  "AG"   "GT"   "CA"   "TG"   "GT"   "TG"   "CA"
```

We can see that the genotypes are identical to ones we imported previously, as should be the case:

```
> g0 == g1

          rs1001 rs6679 rs2401 rs123 rs6679 rs876 mitoA1
cd289982    TRUE   TRUE   TRUE  TRUE   TRUE    NA   TRUE
cd325285    TRUE   TRUE   TRUE  TRUE   TRUE    NA   TRUE
cd357273    TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
cd872422    TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
cd1005389   TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
```

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 rs123 rs6679 rs876 mitoA1
cd289982  "A/A"  "G/T"  "A/A"  "T/T" "G/T"  NA    "A/A"
cd325285  "A/G"  "G/G"  "C/C"  "T/G" "G/G"  NA    "C/C"
cd357273  "A/G"  "G/G"  "C/C"  "T/T" "G/G"  "T/T" NA
cd872422  "A/A"  "G/T"  "C/C"  "T/T" "G/T"  "G/G" NA
cd1005389 "G/G"  "G/G"  "C/A"  "T/T" "G/G"  "T/T" NA

> as.character(df@gtdata@strand)

rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
   "+"    "+"    "+"    "+"    "+"    "+"    "+"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
  "AG"   "GT"   "CA"   "TG"   "GT"   "TG"   "CA"
```

We can see that the genotypes are identical to ones we imported previously, as should be the case:

```
> g0 == g1

          rs1001 rs6679 rs2401 rs123 rs6679 rs876 mitoA1
cd289982    TRUE   TRUE   TRUE  TRUE   TRUE    NA   TRUE
cd325285    TRUE   TRUE   TRUE  TRUE   TRUE    NA   TRUE
cd357273    TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
cd872422    TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
cd1005389   TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
```

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 rs123 rs6679 rs876 mitoA1
cd289982  "A/A"  "G/T"  "A/A"  "T/T" "G/T"  NA    "A/A"
cd325285  "A/G"  "G/G"  "C/C"  "T/G" "G/G"  NA    "C/C"
cd357273  "A/G"  "G/G"  "C/C"  "T/T" "G/G"  "T/T" NA
cd872422  "A/A"  "G/T"  "C/C"  "T/T" "G/T"  "G/G" NA
cd1005389 "G/G"  "G/G"  "C/A"  "T/T" "G/G"  "T/T" NA
```

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

rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
   "+"    "+"    "+"    "+"    "+"    "+"    "+"

> as.character(df@gtdata@coding)

rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
  "AG"   "GT"   "CA"   "TG"   "GT"   "TG"   "CA"

> g0 == g1

          rs1001 rs6679 rs2401 rs123 rs6679 rs876 mitoA1
cd289982    TRUE   TRUE   TRUE  TRUE   TRUE    NA   TRUE
cd325285    TRUE   TRUE   TRUE  TRUE   TRUE    NA   TRUE
cd357273    TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
cd872422    TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
cd1005389   TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
```

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 rs123 rs6679 rs876 mitoA1
cd289982  "A/A"  "G/T"  "A/A"  "T/T" "G/T"  NA    "A/A"
cd325285  "A/G"  "G/G"  "C/C"  "T/G" "G/G"  NA    "C/C"
cd357273  "A/G"  "G/G"  "C/C"  "T/T" "G/G"  "T/T" NA
cd872422  "A/A"  "G/T"  "C/C"  "T/T" "G/T"  "G/G" NA
cd1005389 "G/G"  "G/G"  "C/A"  "T/T" "G/G"  "T/T" NA
```

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

```
rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
   "+"    "+"    "+"    "-"    "-"    "+"    "-"
```

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

```
rs1001 rs6679 rs2401  rs123 rs6679  rs876 mitoA1
  "AG"   "GT"   "CA"   "TG"   "GT"   "TG"   "CA"
```

```
> g0 == g1
```

```
          rs1001 rs6679 rs2401 rs123 rs6679 rs876 mitoA1
cd289982    TRUE   TRUE   TRUE  TRUE   TRUE    NA   TRUE
cd325285    TRUE   TRUE   TRUE  TRUE   TRUE    NA   TRUE
cd357273    TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
cd872422    TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
cd1005389   TRUE   TRUE   TRUE  TRUE   TRUE  TRUE     NA
```

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

Appendix B

Answers to exercises

B.0.1 Exercise 2.1:

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.0.2 Exercise 2.2:

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.0.3 Exercise 2.3:

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
```

B.0.4 Exercise 2.3:

```
> 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 B.1) 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.0.5 Exercise 1:

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

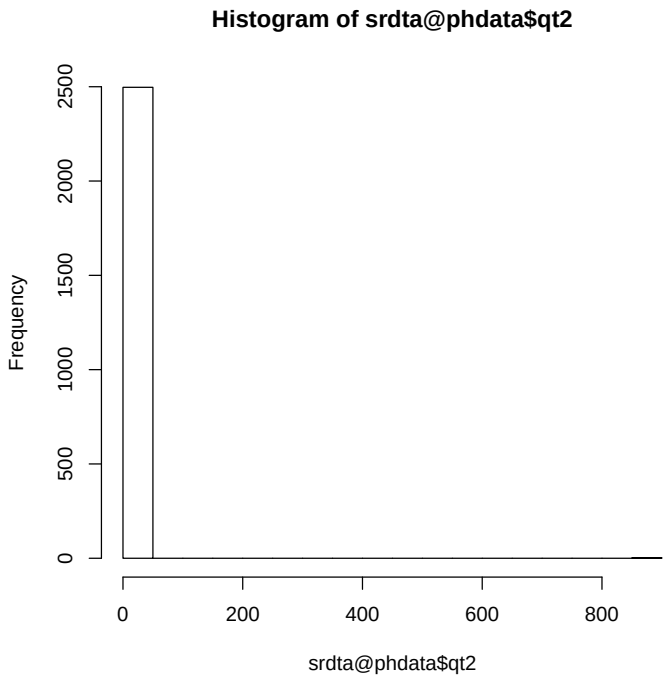

Figure B.1: Histogram of qt2

B.0.6 Exercise 2:

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.0.7 Exercise 3:

How many cases and controls are present?

```
> table(aff)

aff
   0    1
1951  549
```

There are 549 cases and 1951 controls.

B.0.8 Exercise 4:

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.0.9 Exercise 5:

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.0.10 Exercise 6:

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.0.11 Exercise 7:

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.0.12 Exercise 8:

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.0.13 Exercise 9:

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.0.14 Exercise 10:

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 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 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.0.15 Exercise 11:

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.0.16 Exercise 12:

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.0.17 Exercise 13:

If you adjust the analysis under additive model (question 12) for significant covariates which you discovered in questions 8 and 9, 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.0.18 Exercise 14:

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.0.19 Exercise 4.1:

```
> 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[1]:

```
> 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.0.20 Exercise 4.1:

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.0.21 Exercise 4.2:

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":

[1] 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.0.22 Exercise 4.3:

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  1.281239e-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
```

B.0.23 Exercise 37:

Perform meta-analysis of the data presented in table 10.2. Which allele is the risk one? Is this risk significant? What is pooled Odds Ratio and 95% confidence interval? Do analysis using at least two methods. Which method is better (best) in this situation? Why?

We first need to unify Odds Ratios by using the same effective allele. Let that be the "risk" allele, as may be guessed from a glance to the data, namely "Pro".

When the effects are reported for the other, "Ala" allele, the corresponding ORs for the "Pro" allele can be found using simple relation $OR_{Pro} = 1/OR_{Ala}$.

Thus, the vector of Odds Ratios for "Pro" allele is

```
> or.pro <- c(1/0.67, 0.93, 1.08, 1/0.83, 1.22, 1.23)
> or.pro

[1] 1.492537 0.930000 1.080000 1.204819 1.220000 1.230000
```

The corresponding $P - values$ are

```
> p <- c(0.013, 0.6, 0.84, 0.4, 0.25, 0.07)
```

Let us find log-ORs

```
> logor.pro <- log(or.pro)
> logor.pro

[1]  0.40047757 -0.07257069  0.07696104  0.18632958  0.19885086  0.20701417
```

Corresponding squared standard errors are

```
> s2 <- logor.pro * logor.pro/qchisq(1 - p, 1)
> s2

[1] 0.02599764 0.01915121 0.14531060 0.04901514 0.02988102 0.01305349
```

and weights are

```
> w <- 1/s2
> w

[1] 38.46503 52.21601  6.88181 20.40186 33.46606 76.60788
```

Thus the pooled estimate of log-OR is

```
> p.logor.pro <- sum(w * logor.pro)/sum(w)
> p.logor.pro

[1] 0.1686548
```

and the standard error is

```
> p.s <- 1/sqrt(sum(w))
> p.s

[1] 0.066221
```

Thus the pooled estimate of Odds Ratio from association between type 2 diabetes and "Ala" allele is

```
> exp(p.logor.pro)

[1] 1.183711
```

and the 95% confidence interval is

```
> exp(p.logor.pro - 1.96 * p.s)

[1] 1.039627

> exp(p.logor.pro + 1.96 * p.s)

[1] 1.347765
```

The χ^2 test for association and corresponding $P - value$ are

```
> p.chi2 <- (p.logor.pro/p.s)^2
> p.chi2

[1] 6.486429

> p.pval <- 1 - pchisq(p.chi2, 1)
> p.pval

[1] 0.01087011
```

Z-score pooling though may be more appropriate method for such differentially designed studies (e.g. control groups are very different). To get Z-score pooling working, we need first find Z-scores from P-values

```
> p <- c(0.013, 0.6, 0.84, 0.4, 0.25, 0.07)
> z <- sqrt(qchisq(1 - p, 1))
> z

[1] 2.4837693 0.5244005 0.2018935 0.8416212 1.1503494 1.8119107
```

and assign the right sign (let "+" is for the risk effect of "Pro").

```
> effsig <- c(1, -1, 1, 1, 1, 1)
> z <- z * effsig
> z

[1]  2.4837693 -0.5244005  0.2018935  0.8416212  1.1503494  1.8119107
```

Now, we need to assign weights to the studies as

```
> n <- c(221, 306, 71, 164, 242, 471)
> w <- sqrt(n)
```

ane the pooled estimate of Z and corresponding $P - value$ are

```
> zpoo <- sum(w * z)/sqrt(sum(w^2))
> zpoo

[1] 2.537333

> 1 - pchisq(zpoo * zpoo, 1)

[1] 0.01117008
```

As you can see the results are almost identical to the previous obtained with inverse variance pooling.

B.0.24 Exercise 38:

Perform meta-analsys excluding the original report (study 1). Is there still significant association between Pro12Ala and diabetes?

The answer to this exercise can be obtained in exactly the same manner, as for the previous one, limiting our consideration to the last five studies.

Thus, the vector of Odds Ratios for "Pro" allele is

```
> or.pro <- c(0.93, 1.08, 1/0.83, 1.22, 1.23)
> or.pro

[1] 0.930000 1.080000 1.204819 1.220000 1.230000
```

The corresponding $P - values$ are

```
> p <- c(0.6, 0.84, 0.4, 0.25, 0.07)
```

Let us find log-ORs

```
> logor.pro <- log(or.pro)
> logor.pro

[1] -0.07257069  0.07696104  0.18632958  0.19885086  0.20701417
```

Corresponding squared standard errors are

```
> s2 <- logor.pro * logor.pro/qchisq(1 - p, 1)
> s2

[1] 0.01915121 0.14531060 0.04901514 0.02988102 0.01305349
```

and weights are

```
> w <- 1/s2
> w

[1] 52.21601  6.88181 20.40186 33.46606 76.60788
```

Thus the pooled estimate of log-OR is

```
> p.logor.pro <- sum(w * logor.pro)/sum(w)
> p.logor.pro

[1] 0.1216172
```

and the standard error is

```
> p.s <- 1/sqrt(sum(w))
> p.s

[1] 0.07262917
```

Thus the pooled estimate of Odds Ratio from association between type 2 diabetes and "Ala" allele is

```
> exp(p.logor.pro)
```

```
[1] 1.129322
```

and the 95% confidence interval is

```
> exp(p.logor.pro - 1.96 * p.s)
```

```
[1] 0.9794776
```

```
> exp(p.logor.pro + 1.96 * p.s)
```

```
[1] 1.302090
```

The χ^2 test for association and corresponding $P - value$ are

```
> p.chi2 <- (p.logor.pro/p.s)^2
> p.chi2
```

```
[1] 2.803937
```

```
> p.pval <- 1 - pchisq(p.chi2, 1)
> p.pval
```

```
[1] 0.09403318
```

Z-score pooling though may be more appropriate method for such differentially designed studies (e.g. control groups are very different). To get Z-score pooling working, we need first find Z-scores from P-values

```
> p <- c(0.6, 0.84, 0.4, 0.25, 0.07)
> z <- sqrt(qchisq(1 - p, 1))
> z
```

```
[1] 0.5244005 0.2018935 0.8416212 1.1503494 1.8119107
```

and assign the right sign (let "+" is for the risk effect of "Pro").

```
> effsig <- c(-1, 1, 1, 1, 1)
> z <- z * effsig
> z
```

```
[1] -0.5244005  0.2018935  0.8416212  1.1503494  1.8119107
```

Now, we need to assign weights to the studies as

```
> n <- c(306, 71, 164, 242, 471)
> w <- sqrt(n)
```

ane the pooled estimate of Z and corresponding $P - value$ are

```
> zpoo <- sum(w * z)/sqrt(sum(w^2))
> zpoo
```

```
[1] 1.709151
```

```
> 1 - pchisq(zpoo * zpoo, 1)
```

```
[1] 0.08742297
```

As you can see the results are almost identical to the previous obtained with inverse variance pooling.

Appendix C

Unsorted answers to exercises section 3

Answer of Exercise 1

A-ba-ba!

Answer of Exercise 2

Try `help.search("Fisher")` and `help.search("Student t-test")`. You will find that the corresponding functions are `fisher.test` `t.test`.

Q.1 : How many SNPs are described in this data frame?

```
> attach(popdat)

        The following object(s) are masked from popdat ( position 3 ) :

         aff qt sex snp1 snp10 snp2 snp3 snp4 snp5 snp6 snp7 snp8 snp9 subj


        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

Q.2 : 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.

Q.3 : How many cases and controls are present?

```
> table(aff)

aff
   0    1
1951  549
```

There are 549 cases and 1951 controls.

Q.4 : 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.

Q.5 : 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.

Q.6 : 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 $r^2 \geq 0.8$.

Q.7 : 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.

Q.8 : 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.

Q.9 : 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.

Q.10 : 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 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 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 $P \leq 0.05$.

Q.11 : 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.

Q.12 : 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.

Q.13 : If you adjust the analysis under additive model (question 12) for significant covariates which you discovered in questions 8 and 9, 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.

Q.14 : 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.

Bibliography

Index