GenABEL tutorial

May 16, 2013

Contents

Chapter 1

Overview

This introduction is outdated: now the `GenABEL-package` is the project, the suite, and the package, see http://www.genabel.org/developers

`GenABEL-package` is an R library developed to facilitate Genome-Wide Association (GWA) analysis of binary and quantitative traits. `GenABEL-package` 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-package`, it is highly recommended as this knowledge will improve flexibility and quality of your analysis.

Originally `GenABEL-package` 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-package` 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-package` can be found at the web site http://www.genabel.org.

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

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

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

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

The section 7 ("GWA in presence of genetic stratification: practice") is dedicated to analysis in the presence of population stratification and analysis of family-based data.

Genetic data imputations are covered in the section **??**, "**??**".

The last section, 11 ("Analysis of selected region"), is dedicated to analysis of haplotype association and analysis of SNP interactions.

Information on importing the data from different file formats to `GenABEL-package` is given in appendix A ("Importing data to `GenABEL-package`"). Answers to exercises are provided at the end of the respective chapters.

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

1.1 Download necessary files

This code needs to be run prior to other parts of tutorial. We reccommend that prior to any actions you create a new directory, say, 'exercisesGenABEL', to keep all of your working tutorial files there.

Start R and make sure that your working directory is set to a proper location. Your current working directory can be queried by command 'getwd()'. Use 'setwd' command to set the working directory.

The next lines of code kill the 'RData' directory if it is present in your working directory (danger! danger!) to make new clean data installation. Paste this code into R:

```
unlink("RData",recursive=TRUE,force=TRUE)
dir.create("RData")
```

Now, fetch the necessary data from the server. First, define the download procedure

```
myDownloads <- function(baseUrl,baseLocal,files) {
for (cFile in files) {
    cFileUrl <- paste(baseUrl,cFile,sep="")
    cFileLocal <- paste(baseLocal,cFile,sep="")
    tryDownload <- try(
                       download.file(url=cFileUrl,destfile=cFileLocal)
                       )
    if ( is(tryDownload,"try-error") )
      stop(paste("can not download",cFileUrl,"into",cFileLocal,":",tryDownload))
  }
}
```

Second, download data files:

```
baseUrl <- "http://www.genabel.org/sites/default/files/data/"
baseLocal <- "RData/"
dataFiles <- c(
           "assocbase.RData",
           "popdat.RData",
           "mach1.out.mlinfo",
           "mach1.mldose.fvi",
           "mach1.mldose.fvd",
           "rcT.PHE",
           "gen0.illu",
           "gen0.illuwos",
           "gen0.tped",
           "gen0.tfam",
           "gen0.ped",
           "map0.dat",
           "emap0.dat",
           "phe0.dat",
           "ImputedDataAnalysis.RData")
myDownloads(baseUrl,baseLocal,dataFiles)
```

That's it! - now you are fully set to start with the `GenABEL` tutorial!

Chapter 2

Introduction to R

In this section we will consider the basic R data types and operations, as well as tools for the 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 one. 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 an introduction to regression analysis.

2.1 Basic R data types and operations

In contrast with many other statistical analysis packages, analysis in R is not based on a graphical 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 your default 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, we have rounded the result to the two digits after the floating point – just 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 a name say '`fun`', help may be obtained by typing '`help(fun)`' (or `?fun`) on the command line.

R help pages have a 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.

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. E.g. try '`example(log10)`'.

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

Exercise 1. Explore help for Wilcoxon test

Explore the help page for the Wilcoxon test (function: `wilcox.test`) and answer the following questions:

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

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. Finding functions and help pages

Try to find out what are the functions to do the

1. Fisher exact test
2. T-test

One of the important R operations is *assignment*, which is done with the '`<-`' 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 the name of an already existing variable or an expression. For example, we if want to assign the value '2' to the variable '`a`', and value '3' to the variable '`b`' we would use the assignment operator in the following way:

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

Typing the variable name on the 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 earlier using numerical arguments.

While the variables 'a' and 'b' contain single numeric values, variables in general can be multi-dimensional; a one-dimensional example of such is the vector (or 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] 1.00000000 0.33333333 0.20000000 0.14285714 0.09090909

> 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 elements of a vector, use

```
> sum(v)

[1] 27
```

Other examples of such functions are `length`, returning the 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 operation which helps 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, the operation `c(2:4)` stands 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 the very attractive features of R data objects is the 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 should 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 want to see the heights in people that 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"
```

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

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

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

[1] 2 3 4

> tmpids[vec]

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

You can see that now `vec` contains a vector whose elements are the indices of the elements of `tmphgt` for which the logical condition holds.

Sub-setting for 2D objects (matrices) is done in a 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 element 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 2×2 set of elements contained in upper left corner, you can do

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

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

[3]Because it treats NAs for you

	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.

Or you can even get the variables that 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, a complete vector is returned for this dimension. For example, here we retrieve the first row

```
> 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
```

Another way to address elements of a matrix is to use a one-dimensional index. For example, if you want to access the 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 represented as a vector whose elements are numbered consecutively: 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 element in the bottom-right corner has the maximal value, as shown in Table 2.1.

You can sub-set matrices using logical conditions or indexes like you can with vectors. For example, if we want to see which 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 indicess of the matrix, is returned. This vector indicates the elements 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 is fulfilled either by

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

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

or using

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

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

Once again, the latter method should be preferred. Consider the example where some elements of the matrix are missing (`NA`) – a situation which is common in real data analysis. Let us replace element 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`). Sometimes you need to figure out what the class of a certain object is. 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 elements, belonging to some lower-level (e.g. character/numeric) class. To see what is the class of the matrix elements, try

```
> a[1, ]

[1] 11 21 31

> class(a[1, ])

[1] "numeric"
```

which says that elements (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"    "old"    "ov"     "tmphgt" "tmpids" "v"
[9] "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 took a long time to create. Sometimes 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 the `tmphgt` and `tmpids` variables you can use

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

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

```
> ls()

[1] "a"   "b"   "dje" "old" "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 via the `help.start()` command. To search help for some keyword `keywrd`, you can use the `help.search(keywrd)` command. To get a 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 an 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.
- The function `which(A)` returns the index of the elements of A which are `TRUE`
- You can see the objects available in your workspace by using the `ls()` command
- Unnecessary objects (say, `tmphgt`) can be deleted from the workspace using the `rm` command, e.g. `rm(tmphgt)`
- You can leave R using the `q()` command

Exercise 3. Exploring srdta

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

```
> library(GenABEL)
```

and load the data by

```
> data(srdta)
```

The vector containing the study subjects' sex can be accessed through `male(srdta)`; this vector's value is 1 when the corresponding person is male and 0 otherwise. The vector containing SNP names can be accessed via `snpnames(srdta)`, chromosome ID – through `chromosome(srdta)` and map – through `map(srdta)`. 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 22^{nd} person?
3. How many males are observed among the first hundred subjects?
4. How many FEMALES are among the 4^{th} hundred?
5. What is the male proportion in the first 1000 people?
6. What is the FEMALE proportion in second 1000 (1001:2000) people?
7. What is name, chromosome and map position of 33^{rd} marker?
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 the example data set `assoc`. Start R with a 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)

[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 the names are of the 7 variables present in the data frame. To see what the variable names are, 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, logical), 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 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0
 [38] 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0
 [75] 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0
[112] 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1
[149] 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1
[186] 1 1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0
[223] 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0
```

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

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

Exercise 4. Exploring `assoc`

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

A 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 a 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] 75

> 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   75   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     5   0   0  0.1009220  A/B  B/A  B/A
6     6   1   0 -0.1724321  A/B  A/A  A/A
7     7   0   0 -0.3378473  B/B  A/A  A/A
8     8   0   0 -1.7112925  A/A  B/B <NA>
9     9   1   0 -0.4815822  A/B  B/A  B/A
10   10   1   0  1.2281232  A/A  B/B  B/B
11   11   0   0  0.5993945  A/B  B/A  B/A
12   12   0   0  1.9792190  A/A  B/B  B/B
13   13   1   0  1.5435921  A/A  B/B  B/B
14   14   0   0 -1.6242738  A/B  B/A  B/A
15   15   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 ranging 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 `qt` values over 1.4, you can find out what the indices of these people are:

```
> 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 complete data with

```
> assoc$subj[vec]

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

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

```
> assoc$subj[vec]

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

Or, if we are interested to find what the IDs and the SNP genotypes are of these people, we can try

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

    subj snp4 snp5 snp6
12    12  A/A  B/B  B/B
13    13  A/A  B/B  B/B
33    33  A/A  B/B  B/B
41    41  A/A  B/A  B/A
54    54  A/B  B/A  B/A
68    68  A/A  B/B  B/B
72    72  A/A  B/A  B/A
76    76  A/B  B/A  B/A
89    89  A/A  B/B  B/B
106  106  A/B  B/A  B/A
118  118  A/B  B/A  B/A
142  142  A/B  B/A  B/A
156  156  A/A  B/B  B/B
```

```
161  161  A/B  B/A  B/A
175  175  A/B  B/A  B/A
181  181  A/B  B/A  B/A
193  193  A/A  B/B  B/B
219  219  A/B  B/A  B/A
241  241  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 the first, fifth, sixth and seventh column (variable).

The same result can be obtained using variable names instead of the variables' indices. To remind you the variable names can be found with:

```
> names(assoc)
```

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

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

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

```
    subj snp4 snp5 snp6
12    12  A/A  B/B  B/B
13    13  A/A  B/B  B/B
33    33  A/A  B/B  B/B
41    41  A/A  B/A  B/A
54    54  A/B  B/A  B/A
68    68  A/A  B/B  B/B
72    72  A/A  B/A  B/A
76    76  A/B  B/A  B/A
89    89  A/A  B/B  B/B
106  106  A/B  B/A  B/A
118  118  A/B  B/A  B/A
142  142  A/B  B/A  B/A
156  156  A/A  B/B  B/B
161  161  A/B  B/A  B/A
175  175  A/B  B/A  B/A
181  181  A/B  B/A  B/A
193  193  A/A  B/B  B/B
219  219  A/B  B/A  B/A
241  241  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] 75
```

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 the `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 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 the `detach()` function to remove a certain data frame from the search path, e.g. after

```
> detach(assoc)
```

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

```
> assoc$subj[75]

[1] 75
```

Summary:

- The list of available objects can be viewed with `ls()`; the class of some object `obj` can be examined with `class(obj)`.
- Simple summary statistics for numeric variables can be generated by using the `summary` function
- A 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 a 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 another data frame), use `detach(frame)`.

Exercise 5. Explore the phenotypic part of `srdta`

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 `phdata(srdta)`. Explore this data frame and answer the questions

1. What is the value of the 4^{th} variable for subject number 75?

[5] and also a matrix

2. What is the value of variable 1 for person 75? Check the value of this variable for the first ten people. Can you guess what the 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 total number of males, you can use

```
> sum(sex==1)

[1] 129
```

and to determine what is the proportion of males 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 the 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 elements of `qt`. The `sum()` function of R precisely does the operation we need. However, if we try it

```
> sum(qt)

[1] -29.79333
```

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

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

[1] -29.79333
```

where `na.rm=TRUE` tells R that missing variables should be be removed (Non-Available.ReMove=True)[6].

We can now try to compute the mean with

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

[1] -0.1191733
```

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 elements in `qt` that are not missing.

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

```
> 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 FALSE FALSE FALSE FALSE
 [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [49] FALSE FALSE 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 FALSE 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 FALSE 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 FALSE 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 0 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:

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

```
> !is.na(qt)

  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [76] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [91] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[106] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[121] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[136] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[151] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[166] TRUE TRUE TRUE 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 TRUE TRUE TRUE
[196] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[211] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[226] TRUE TRUE TRUE 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] 250
```

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

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

[1] -0.1191733
```

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

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

[1] -0.1191733
```

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

```
> table(sex)

sex
  0   1
121 129
```

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

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

Tables of other qualitative variables, such as affection status 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
98 31

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

 0  1
96 25
```

On the 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

Exercise 6. Explore `assoc`

Explore the phenotypic variables present 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 96 25
  1 98 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 96 25
  1 98 31
```

and this object may be analysed further.

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

```
> a[2, 2]

[1] 31
```

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

```
> prop.table(a)

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

Needless to say, this is equivalent to

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

        0     1
  0 0.384 0.100
  1 0.392 0.124
```

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

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

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

telling us that 55.4% of affected individuals are male.

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

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

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

telling us that 55.4% of male are affected.

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

```
> fisher.test(a)

        Fisher's Exact Test for Count Data

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

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

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

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

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

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

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

showing that proportion of cases is similar in the '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 doing a χ^2 test

```
> chisq.test(a)

        Pearson's Chi-squared test

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

which gives a (insignificant) genotypic association test with 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-package` 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 the `GenABEL-package`. As you can remember from an earlier exercise, that library is loaded with `library(GenABEL)` and the data are loaded with `data(srdta)`. Then the phenotypic data frame may be accessed through `phdata(srdta)`.

Exercise 7. Explore phenotypes in `srdta`

Explore the phenotypic data content of `srdta` object (`phdata(srdta)`).

1. How many observations and variables are present in the data frame?
2. What are the classes of these variables?

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

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

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

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

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

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

The resulting histogram is shown in Figure 2.1.

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

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

The resulting plot is shown in Figure 2.2.

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

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

In `GenABEL-package`, there is a special function designed to facilitate phenotypic quality control. This function takes the 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

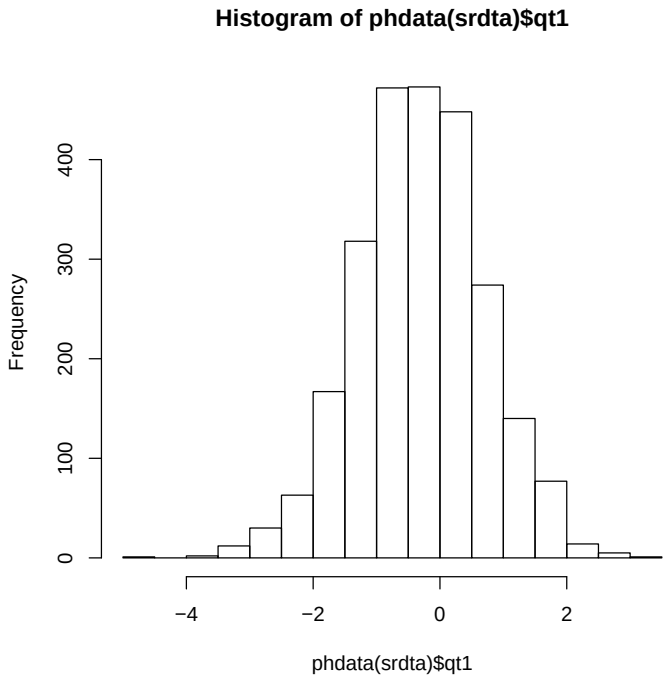

Figure 2.1: Histogram of `qt1`.

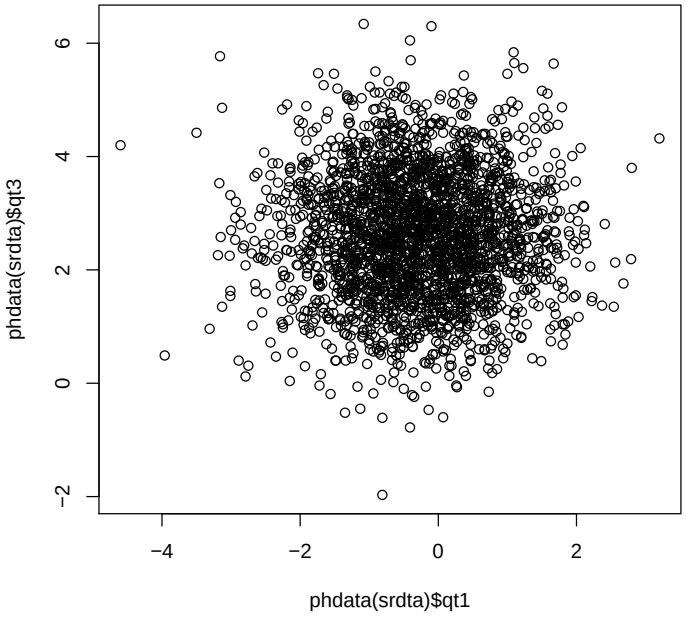

Figure 2.2: Scatter-plot of `qt1` against `qt3`.

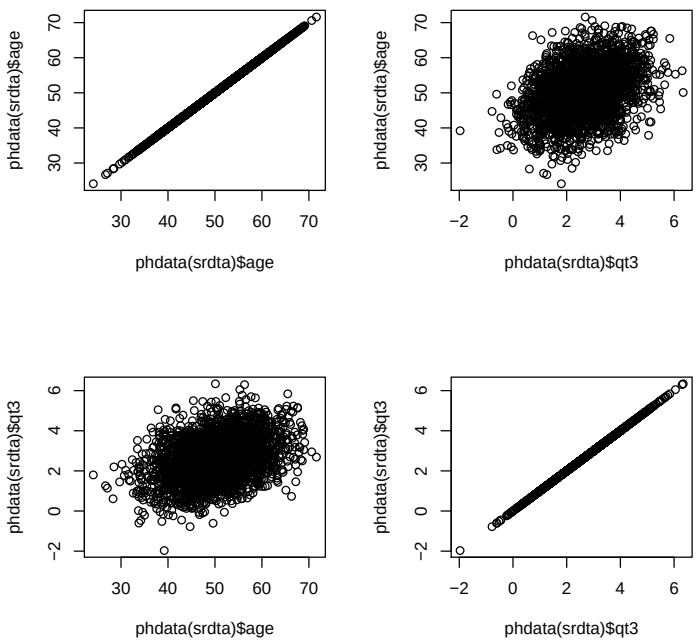

Figure 2.3: Quality control graphs for `sex, age, qt3`

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

--------------------------------
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 shown in figure 2.3.

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

a good idea!

Exercise 8. Exploring the phenotypic part of `srdta`

Explore the `phdata` part of the `srdta` object

1. How many people have age over 65 years?
2. What is the sex distribution (proportion of males) in the people over 65 years old?
3. What are the mean, median, minimum and maximum ages in the sample?
4. Compare the distribution of `qt3` in people younger and older than 65 years. Use the function `sd(A)` to get standard deviation of A.
5. Produce distributions of different traits. Do you see something special?
6. What is the 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 variables, a more general framework allowing investigation of the relation of an outcome to multiple predictors is regression. In R, the function `lm` implements linear regression modelling, and the function `glm` implements generalised linear regression. In this section, we will use these two functions to analyse quantitative and binary outcomes.

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

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

The results of this analysis are stored in object 'a', which has class 'lm' and contains many 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 = phdata(srdta)$qt2 ~ phdata(srdta)$age + phdata(srdta)$sex)

Residuals:
   Min     1Q Median     3Q    Max
 -5.65  -1.80  -1.03  -0.31 883.08

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       -1.55892    4.41667  -0.353    0.724
phdata(srdta)$age  0.14022    0.08668   1.618    0.106
phdata(srdta)$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 `phdata(srdta)` before every trait name, you may attach the data to the search path:

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

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
 -5.65  -1.80  -1.03  -0.31 883.08

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] 1.4617
```

2.5 Answers to exercises

Answer of Exercise 1. Explore help for Wilcoxon test

By default (if 'exact' is not specified), an exact p-value is computed if the samples contain less than 50 finite values and there are no ties. Otherwise, a normal approximation is used.

Answer of Exercise 2. Finding functions and help pages

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

Answer of Exercise 3. Exploring srdta

For the first person the id is "p1" and the sex code is 1 (1=male, 0=female)

```
> idnames(srdta)[1]
```

```
[1] "p1"
```

```
> male(srdta)[1]
```

```
p1
 1
```

The id for the 22nd person is "p22" and sex code is 1:

```
> idnames(srdta)[22]
```

```
[1] "p22"
```

```
> male(srdta)[22]
```

```
p22
  1
```

Among the first 100 subjects, there are 53 males:

```
> sum(male(srdta)[1:100])
```

```
[1] 53
```

Among the 4th hundred subjects there are 45 females:

```
> 100 - sum(male(srdta)[301:400])
```

```
[1] 45
```

The male proportion among the first 1000 people is

```
> mean(male(srdta)[1:1000])
```

```
[1] 0.508
```

The female proportion among the second 1000 people is

```
> 1 - mean(male(srdta)[1001:2000])
```

```
[1] 0.476
```

Name, chromosome and map position of the 33rd marker are:

```
> snpnames(srdta)[33]
```

```
[1] "rs422"
```

```
> chromosome(srdta)[33]
```

```
rs422
  "1"
```

```
> map(srdta)[33]
```

```
 rs422
105500
```

The map positions for and distance between markers 25 and 26 are:

```
> pos25 <- map(srdta)[25]
> pos25

rs365
91250

> pos26 <- map(srdta)[26]
> pos26

rs372
92750

> pos26 - pos25

rs372
 1500
```

Answer of Exercise 4. Exploring `assoc`

Here is an script which automatically explores the classes of variables in `assoc`:

```
> for (i in names(assoc)) {
+   cat("Variable '", i ,"' has class '", class(assoc[, i]), "'\n", sep="")
+ }

Variable 'subj' has class 'integer'
Variable 'sex' has class 'numeric'
Variable 'aff' has class 'numeric'
Variable 'qt' has class 'numeric'
Variable 'snp4' has class 'genotypefactor'
Variable 'snp5' has class 'genotypefactor'
Variable 'snp6' has class 'genotypefactor'
```

In this so-called for-loop the variable `i` cycles through all names in `assoc` and for each of them it uses the `cat` function to print the name of the variable and its class. The `\n` is the code for a new line.

Answer of Exercise 5. Explore the phenotypic part of `srdta`

Load the data and look at the few first rows of the phenotypic data frame:

```
> data(srdta)
> phdata(srdta)[1:5, ]

   id sex  age   qt1    qt2  qt3 bt
p1 p1   1 43.4 -0.58   4.46 1.43  0
p2 p2   1 48.2  0.80   6.32 3.90  1
p3 p3   0 37.9 -0.52   3.26 5.05  1
p4 p4   1 53.8 -1.55 888.00 3.76  1
p5 p5   1 47.5  0.25   5.70 2.89  1
```

The value of the 4^{th} variable of person 75:

```
> phdata(srdta)[75, 4]

[1] -0.04
```

The value for variable 1 is

```
> phdata(srdta)[75, 1]

[1] "p75"
```

Also, if we check the first 10 elements we see

```
> phdata(srdta)[1:10, 1]

 [1] "p1"  "p2"  "p3"  "p4"  "p5"  "p6"  "p7"  "p8"  "p9"  "p10"
```

This is the individual ID.

The sum for variable 2 is

```
> sum(phdata(srdta)[, 2])

[1] 1275
```

This is the sex variable – so there are 1275 males in the data set.

Answer of Exercise 6. Explore `assoc`

The number of affected (coded with '1') and unaffected ('0') is

```
> table(aff)

aff
  0   1
194  56
```

The proportion of unaffected and affected is

```
> prop.table(table(aff))

aff
    0     1
0.776 0.224
```

Distribution of the 'snp4' is

```
> t <- table(snp4)
> t

snp4
A/A A/B B/B
109 105  29

> prop.table(t)

snp4
      A/A       A/B       B/B
0.4485597 0.4320988 0.1193416
```

Answer of Exercise 7. Explore phenotypes in `srdta`

Number of people:

```
> nids(srdta)
```

```
[1] 2500
```

Number of variables:

```
> length( names( phdata(srdta) ) )
```

```
[1] 7
```

The same – dimensions of phenotypic data frame:

```
> dim(phdata(srdta))
```

```
[1] 2500    7
```

The class of variables in phenotypic data frame:

```
> for (i in names(phdata(srdta))) {
+   cat("The class of variable '", i ,"' is '",
+       class(phdata(srdta)[, i]), "'\n", sep="")
+ }
```

```
The class of variable 'id' is 'character'
The class of variable 'sex' is 'integer'
The class of variable 'age' is 'numeric'
The class of variable 'qt1' is 'numeric'
The class of variable 'qt2' is 'numeric'
The class of variable 'qt3' is 'numeric'
The class of variable 'bt' is 'integer'
```

Answer of Exercise 8. Exploring the phenotypic part of `srdta`

To obtain the number of people with age > 65 y.o., you can use any of the following

```
> sum( phdata(srdta)$age > 65 )
```

```
[1] 48
```

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

```
[1] 48
```

To get sex of these people use any of:

```
> sx65 <- phdata(srdta)$sex[ phdata(srdta)$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 <- phdata(srdta)$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 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 males is 0.541666666666667.

The distributions of `qt3` in people younger and older than 65 are:

```
> summary(phdata(srdta)$qt3[vec])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.730   2.690   3.480   3.499   4.265   5.840

> sd( phdata(srdta)$qt3[vec], na.rm=TRUE )

[1] 1.128701

> young <- which(phdata(srdta)$age<65)
> summary( phdata(srdta)$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

> sd( phdata(srdta)$qt3[young], na.rm=TRUE )

[1] 1.093374
```

The Mean, median, min and max of age:

```
> summary( phdata(srdta)$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 (you can generate it using `hist(phdata(srdta)$qt2)`): it seems there are a few very strong outliers. You can also see that with `summary`:

```
> summary( phdata(srdta)$qt2 )

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   4.220   5.045   6.122   5.910 888.000
```

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-package`.

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`. **THIS FILE IS THE SAME AS THE assocbase.RData, so I load and patch it below** Load library `genetics`, which we will need for testing HWE and computations of LD by

```
> library(genetics)
```

```
NOTE: THIS PACKAGE IS NOW OBSOLETE.

  The R-Genetics project has developed an set of enhanced genetics
  packages to replace 'genetics'. Please visit the project homepage
  at http://rgenetics.org for informtion.
```

The file you have loaded contains single data frame `assocg`. Let us load

```
> #load("RData/ge03d1p1.RData")
> # load assocbase data and convert snps into proper 'genetics' format
> load("RData/assocbase.RData")
> assocg <- assoc
> assocg$snp4 <- as.genotype(assocg$snp4)
> assocg$snp5 <- as.genotype(assocg$snp5)
> assocg$snp6 <- as.genotype(assocg$snp6)
```

and 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 20). 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 23:

```
> assocg[5:15,]

   subj sex aff         qt snp4 snp5 snp6
5     5   0   0  0.1009220  A/B  B/A  B/A
6     6   1   0 -0.1724321  A/B  A/A  A/A
7     7   0   0 -0.3378473  B/B  A/A  A/A
8     8   0   0 -1.7112925  A/A  B/B <NA>
9     9   1   0 -0.4815822  A/B  B/A  B/A
10   10   1   0  1.2281232  A/A  B/B  B/B
11   11   0   0  0.5993945  A/B  B/A  B/A
12   12   0   0  1.9792190  A/A  B/B  B/B
13   13   1   0  1.5435921  A/A  B/B  B/B
14   14   0   0 -1.6242738  A/B  B/A  B/A
15   15   0   0 -0.5160331  A/A  B/B  B/B
```

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

```
> class(assocg$snp4)
```

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

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

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

```
> attach(assocg)
```

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

```
> summary(snp4)

Number of samples typed: 243 (97.2%)

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


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

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

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

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

Number of samples typed: 190 (97.9%)

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


Genotype Frequency:
    Count Proportion
B/B    22       0.12
```

```
A/B    81       0.43
A/A    87       0.46
NA      4         NA

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

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

Number of samples typed: 53 (94.6%)

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


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

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

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

```
> HWE.exact(snp4)

        Exact Test for Hardy-Weinberg Equilibrium

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

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

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

        Exact Test for Hardy-Weinberg Equilibrium

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

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

```
> LD(snp4,snp5)
```

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

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

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

```
> 0.8683117*0.8683117

[1] 0.7539652
```

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

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

[1] 0.7539652
```

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

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

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

```
> names(ld45)

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

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

```
> ld45$"D'"

[1] 0.9997352
```

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

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

and later check

```
> ldall$"P-value"

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

to see significance,

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

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

for D' and

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

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

for r^2.

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

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

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

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

Resulting plot is shown at figure 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
```

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

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

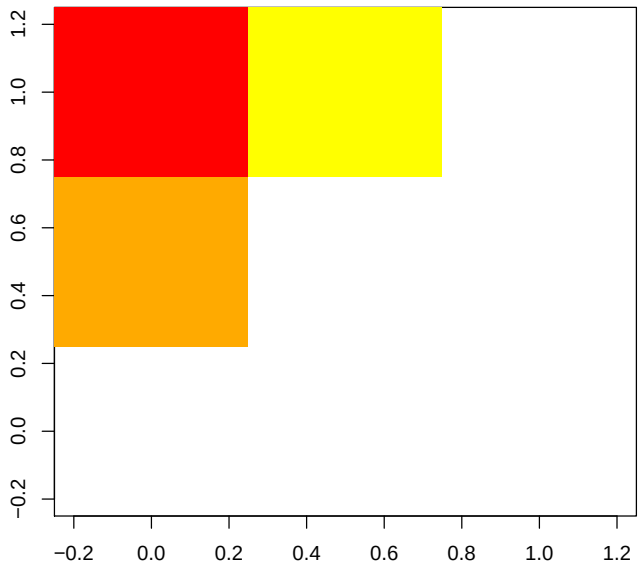

Figure 3.1: r^2 plot for snp4, snp5 and snp6

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

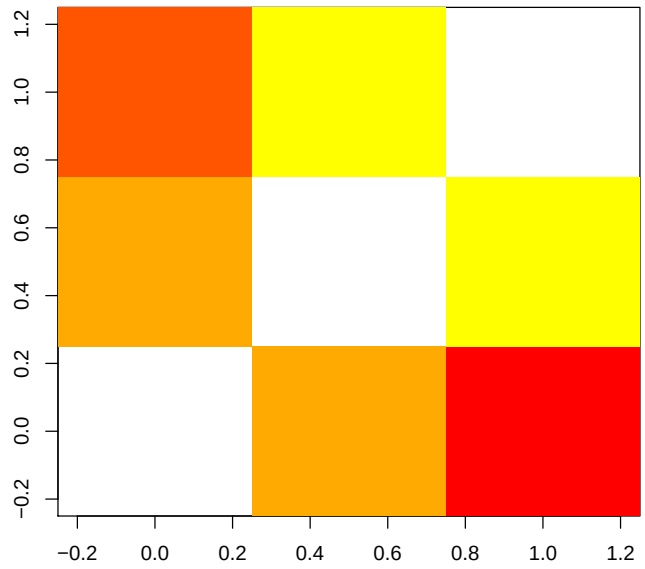

Figure 3.2: r^2 plot for snp4, snp5 and snp6. Above diagonal: LD in cases; below: controls

```
> summary(mg)

Call:
lm(formula = qt ~ snp4)

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

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

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

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

the output) is also not significant.

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

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

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

Residuals:
     Min       1Q   Median       3Q      Max
-2.66442 -0.62417 -0.00875  0.59705  3.08086

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
-2.57049 -0.64596 -0.00264  0.61094  3.01970

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 Pr(>Chi)
NULL                   242     254.91
snp4  2  0.32894       240     254.58   0.8483
```

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
-0.7549  -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: 258.6

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: Exploring genetic data using library `genetics`

Start `R` by double-click over the file `ge03d1p2.RData` (Windows) or by changing to the directory containing the data, starting `R` and loading the data set with `load("ge03d1p2.RData")` (Linux). Explore the data frame present and answer the questions.

Ex. 1 — How many SNPs are described in this data frame?

Ex. 2 — What is the frequency (proportion) of snp1 allele 'A'?

Ex. 3 — What is its frequency of 'A' in affected (aff==1)?

Ex. 4 — How many cases and controls are present in the data set?

Ex. 5 — If all subjects are used to test HWE, are there any SNPs out of HWE at nominal $P \leq 0.05$? Which ones?

Ex. 6 — If only controls are used to test the SNPs which are out of HWE in total sample, are these still out of HWE?

Ex. 7 — Which SNP pairs are in strong LD ($r^2 \geq 0.8$)?

Ex. 8 — For SNPs in strong LD, what is r^2 for separate samples of cases and controls?

Ex. 9 — Is there significant association between affection status and sex? What is P-value for association?

Ex. 10 — Is association between the disease and `qt` significant?

Ex. 11 — Which SNPs are significantly associated with affection status at nominal p-value ≤ 0.05? Use general genotypic (2 d.f.) model.

Ex. 12 — 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?

Ex. 13 — Test for association between the quantitative trait `qt` and SNPs 1-10 using additive model. Which SNPs are associated at nominal $P \leq 0.05$?

Ex. 14 — **OPTIONAL, difficulty is medium, but may be time-consuming.** If you adjust the analysis under additive model for sex, how do the findings change? Before doing the exercise, please check the answer to **previous** exercise – it shows a quick way to do testing for all 10 SNPs.

Ex. 15 — Which SNPs are associated with the quantitative trait `qt` at nominal $P \leq 0.05$ when general genotypic (2 d.f. test) model is used?

Ex. 16 — **ADVANCED:** How can you describe the model of action of the significant SNPs? Test if the data are compatible with additive/dominant/recessive model.

3.6 Answers to exercises

Answer (Ex. 1) — The answer is 10 snps:

```
> attach(popdat)
> names(popdat)
 [1] "subj"  "sex"   "aff"   "qt"    "snp1"  "snp2"  "snp3"  "snp4"  "snp5"
[10] "snp6"  "snp7"  "snp8"  "snp9"  "snp10"
```

Answer (Ex. 2) — The frequency of 'A' in all subjects is 0.73:

```
> 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
```

Answer (Ex. 3) — The frequency of A in affected subjects is 0.7:

```
> 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
```

Answer (Ex. 4) — There are 549 cases and 1951 controls:

```
> table(aff)
aff
   0    1
1951  549
```

Answer (Ex. 5) — Only SNP 1 is out of HWE in the total sample. Here is a sciript testing all SNPs (no need to reproduce that, just check the results):

```
> for (i in 1:10) {
+ snpname <- paste("snp",i,sep="")
```

```
+ cat("HWE P-value for SNP",snpname,"is",HWE.exact(get(snpname))$p.value,"\n")
+ }
HWE P-value for SNP snp1 is 0.01083499
HWE P-value for SNP snp2 is 1
HWE P-value for SNP snp3 is 0.4197772
HWE P-value for SNP snp4 is 0.8960298
HWE P-value for SNP snp5 is 0.2960967
HWE P-value for SNP snp6 is 0.5207056
HWE P-value for SNP snp7 is 0.6284575
HWE P-value for SNP snp8 is 0.1309458
HWE P-value for SNP snp9 is 0.4457363
HWE P-value for SNP snp10 is 0.7897327
```

Answer (Ex. 6) — Yes, SNP one is out of HWE also in controls:

```
> 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
```

Answer (Ex. 7) — SNP pairs 4-5 and 5-6 have $r^2 \geq 0.8$:

```
> 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
```

Answer (Ex. 8) — For controls,

```
> #LD(popdat[aff==0,8:10])$"R^2"
> LD(data.frame(snp4,snp5,snp6)[aff==0,])$"R^2"
     snp4     snp5      snp6
snp4   NA 0.806591 0.6419715
snp5   NA       NA 0.8661005
snp6   NA       NA        NA
```

For cases,

```
> #LD(popdat[aff==1,8:10])$"R^2"
> LD(data.frame(snp4,snp5,snp6)[aff==1,])$"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 *et al.* (2006), are required to contrast LD between sample of unequal size.

Answer (Ex. 9) — There is no significant association:

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

```
   sex
aff   0   1
  0 973 978
  1 260 289
```

```
> fisher.test(t)
```

```
        Fisher's Exact Test for Count Data

data:  t
p-value = 0.3104
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.9107753 1.3430565
sample estimates:
odds ratio
  1.105811
```

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

```
Call:
glm(formula = aff ~ sex, family = binomial())

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.7196  -0.7196  -0.6882  -0.6882   1.7644

Coefficients:
```

```
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.31970    0.06981  -18.90   <2e-16 ***
sex          0.10062    0.09673    1.04    0.298
---
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: 2632.0  on 2499  degrees of freedom
Residual deviance: 2630.9  on 2498  degrees of freedom
AIC: 2634.9

Number of Fisher Scoring iterations: 4
```

Answer (Ex. 10) — There is no significant association:

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

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.7326  -0.7079  -0.7012  -0.6905   1.7675

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.26769    0.04832 -26.238   <2e-16 ***
qt          -0.02514    0.04862  -0.517    0.605
---
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: 2632.0  on 2499  degrees of freedom
Residual deviance: 2631.7  on 2498  degrees of freedom
AIC: 2635.7

Number of Fisher Scoring iterations: 4
```

Answer (Ex. 11) — SNPs 5 and 10 are significantly associated:

```
> for (i in 1:10) {
+ snpname <- paste("snp",i,sep="")
+ cat("\nTesting association between aff and SNP",snpname,":\n")
+ print(anova(glm(aff~get(snpname),family=binomial),test="Chisq"))
+ #print(summary(lm(qt~get(snpname)))$coef)
+ }

Testing association between aff and SNP snp1 :
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2373     2493.4
get(snpname)  2   5.4094      2371     2488.0  0.06689 .
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between aff and SNP snp2 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2373     2485.8
get(snpname)  1  0.29367      2372     2485.5   0.5879

Testing association between aff and SNP snp3 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2377     2503.0
get(snpname)  2   2.6087      2375     2500.4   0.2714

Testing association between aff and SNP snp4 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff
```

```
Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2389     2519.1
get(snpname)  2   5.2755      2387     2513.8  0.07152 .
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between aff and SNP snp5 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2382     2440.4
get(snpname)  2   9.2395      2380     2431.2 0.009855 **
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between aff and SNP snp6 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2379     2498.9
get(snpname)  2   1.7969      2377     2497.1   0.4072

Testing association between aff and SNP snp7 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
```

```
NULL                          2367     2487.9
get(snpname)  2   1.3604      2365     2486.6   0.5065

Testing association between aff and SNP snp8 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2370     2489.4
get(snpname)  2   5.5375      2368     2483.9  0.06274 .
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between aff and SNP snp9 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2360     2476.8
get(snpname)  2   1.1891      2358     2475.6   0.5518

Testing association between aff and SNP snp10 :
Analysis of Deviance Table

Model: binomial, link: logit

Response: aff

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          2383     2475.1
get(snpname)  2   6.7328      2381     2468.4  0.03451 *
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

Answer (Ex. 12) — 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:

```
> summary(glm(aff~snp5*snp10,family=binomial()))

Call:
glm(formula = aff ~ snp5 * snp10, family = binomial())

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9906  -0.7340  -0.6323  -0.5215   2.0310

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      -1.50840    0.08905 -16.938  < 2e-16 ***
snp5A/A          -0.41802    0.19722  -2.120   0.0340 *
snp5B/B           0.33441    0.13360   2.503   0.0123 *
snp10A/B         -0.01403    0.18251  -0.077   0.9387
snp10B/B         -0.14983    0.55277  -0.271   0.7863
snp5A/A:snp10A/B  1.48369    0.32750   4.530 5.89e-06 ***
snp5B/B:snp10A/B  0.12989    0.27441   0.473   0.6360
snp5A/A:snp10B/B  0.82348    0.98963   0.832   0.4053
snp5B/B:snp10B/B -0.28562    1.23104  -0.232   0.8165
---
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: 2282.4  on 2268  degrees of freedom
Residual deviance: 2242.9  on 2260  degrees of freedom
  (231 observations deleted due to missingness)
AIC: 2260.9

Number of Fisher Scoring iterations: 4
```

Answer (Ex. 13) — SNPs 1, 4, 5, 6 and 9 are significantly associated at nominal $P \leq 0.05$. Here is a testing script (no need to reproduce that, just check the results):

```
> for (i in 1:10) {
+ snpname <- paste("snp",i,sep="")
+ cat("\nTesting association between qt and SNP",snpname,":\n")
+ testmodel <- lm(qt~as.numeric(get(snpname)))
+ print(summary(testmodel)$coef)
+ }

Testing association between qt and SNP snp1 :
                            Estimate Std. Error   t value    Pr(>|t|)
```

```
(Intercept)              -0.11874800 0.05260279 -2.257447 0.024070746
as.numeric(get(snpname))  0.08859657 0.03147693  2.814651 0.004923315

Testing association between qt and SNP snp2 :
                            Estimate Std. Error    t value  Pr(>|t|)
(Intercept)               0.09149145  0.1841115  0.4969352 0.6192808
as.numeric(get(snpname)) -0.07749967  0.1806078 -0.4291047 0.6678860

Testing association between qt and SNP snp3 :
                            Estimate Std. Error    t value  Pr(>|t|)
(Intercept)               0.06382773 0.05629376  1.1338331 0.2569789
as.numeric(get(snpname)) -0.02517149 0.02894125 -0.8697443 0.3845280

Testing association between qt and SNP snp4 :
                            Estimate Std. Error   t value   Pr(>|t|)
(Intercept)               0.14005988 0.05612775  2.495377 0.01264938
as.numeric(get(snpname)) -0.07284539 0.02982557 -2.442380 0.01466282

Testing association between qt and SNP snp5 :
                            Estimate Std. Error   t value   Pr(>|t|)
(Intercept)              -0.14350846 0.06620992 -2.167477 0.03029734
as.numeric(get(snpname))  0.07404874 0.02941437  2.517434 0.01188645

Testing association between qt and SNP snp6 :
                            Estimate Std. Error   t value   Pr(>|t|)
(Intercept)              -0.12737489 0.06768096 -1.881990 0.05995937
as.numeric(get(snpname))  0.06724115 0.02924840  2.298969 0.02159304

Testing association between qt and SNP snp7 :
                            Estimate Std. Error   t value  Pr(>|t|)
(Intercept)               0.08884244 0.05475991  1.622399 0.1048511
as.numeric(get(snpname)) -0.03774136 0.03152701 -1.197112 0.2313829

Testing association between qt and SNP snp8 :
                            Estimate Std. Error    t value  Pr(>|t|)
(Intercept)              -0.05214665 0.08116533 -0.6424744 0.5206274
as.numeric(get(snpname))  0.06942327 0.07222881  0.9611576 0.3365710

Testing association between qt and SNP snp9 :
                             Estimate Std. Error   t value     Pr(>|t|)
(Intercept)              -0.2201742 0.07115711 -3.094199 0.0019965937
as.numeric(get(snpname))  0.2110978 0.06112112  3.453761 0.0005625794

Testing association between qt and SNP snp10 :
                            Estimate Std. Error    t value  Pr(>|t|)
(Intercept)              -0.01474695 0.05749473 -0.2564921 0.7975930
as.numeric(get(snpname))  0.03140888 0.04251458  0.7387789 0.4601141
```

Answer (Ex. 14) — Generally, results do not change much: still, SNPs 1, 4, 5, 6 and 9 are significantly associated, and p-values are close to these observed without adjustment For SNPs

```
> for (i in 1:10) {
+ snpname <- paste("snp",i,sep="")
+ cat("\nTesting sex-adjusted association between qt and SNP",snpname,":\n")
+ testmodel <- lm(qt~sex+as.numeric(get(snpname)))
+ print(summary(testmodel)$coef)
+ }
Testing sex-adjusted association between qt and SNP snp1 :
                            Estimate Std. Error    t value    Pr(>|t|)
(Intercept)              -0.12783200 0.05660623 -2.2582672 0.024019551
sex                       0.01766209 0.04061074  0.4349117 0.663666095
as.numeric(get(snpname))  0.08868826 0.03148302  2.8170191 0.004887301

Testing sex-adjusted association between qt and SNP snp2 :
                            Estimate Std. Error    t value  Pr(>|t|)
(Intercept)               0.08169692 0.18525892  0.4409878 0.6592621
sex                       0.01977118 0.04101612  0.4820343 0.6298261
as.numeric(get(snpname)) -0.07770464 0.18063757 -0.4301688 0.6671120

Testing sex-adjusted association between qt and SNP snp3 :
                            Estimate Std. Error    t value  Pr(>|t|)
(Intercept)               0.05518829 0.06028776  0.9154146 0.3600669
sex                       0.01626276 0.04057000  0.4008568 0.6885616
as.numeric(get(snpname)) -0.02491788 0.02895328 -0.8606239 0.3895321

Testing sex-adjusted association between qt and SNP snp4 :
                            Estimate Std. Error    t value   Pr(>|t|)
(Intercept)               0.12611774 0.06040009  2.0880388 0.03690009
sex                       0.02553529 0.04083273  0.6253633 0.53179244
as.numeric(get(snpname)) -0.07227905 0.02984312 -2.4219671 0.01551079

Testing sex-adjusted association between qt and SNP snp5 :
                            Estimate Std. Error    t value   Pr(>|t|)
(Intercept)              -0.15272175 0.06880690 -2.2195704 0.02654187
sex                       0.02009534 0.04076087  0.4930057 0.62205408
as.numeric(get(snpname))  0.07357452 0.02943476  2.4995791 0.01250094

Testing sex-adjusted association between qt and SNP snp6 :
                            Estimate Std. Error    t value   Pr(>|t|)
(Intercept)              -0.14110680 0.07002874 -2.0149841 0.04401871
sex                       0.03117861 0.04077551  0.7646406 0.44456146
as.numeric(get(snpname))  0.06634602 0.02927437  2.2663517 0.02351939

Testing sex-adjusted association between qt and SNP snp7 :
                            Estimate Std. Error   t value  Pr(>|t|)
(Intercept)               0.06697438 0.05879604  1.139097 0.2547782
sex                       0.04174468 0.04087001  1.021401 0.3071689
```

```
as.numeric(get(snpname)) -0.03723286 0.03153066 -1.180846 0.2377825

Testing sex-adjusted association between qt and SNP snp8 :
                           Estimate Std. Error    t value  Pr(>|t|)
(Intercept)              -0.06807682 0.08455606 -0.8051087 0.4208378
sex                       0.02760968 0.04102873  0.6729354 0.5010541
as.numeric(get(snpname))  0.07124407 0.07228780  0.9855614 0.3244491

Testing sex-adjusted association between qt and SNP snp9 :
                           Estimate Std. Error    t value     Pr(>|t|)
(Intercept)              -0.22558672 0.07313030 -3.0847229 0.0020610506
sex                       0.01311497 0.04074702  0.3218632 0.7475848790
as.numeric(get(snpname))  0.21002414 0.06122367  3.4304402 0.0006129721

Testing sex-adjusted association between qt and SNP snp10 :
                           Estimate Std. Error    t value  Pr(>|t|)
(Intercept)              -0.02805887 0.06107457 -0.4594198 0.6459747
sex                       0.02628396 0.04064009  0.6467496 0.5178563
as.numeric(get(snpname))  0.03150834 0.04252005  0.7410231 0.4587525
```

Answer (Ex. 15) — SNPs 1, 4, 5 an 9 are significantly associated at nominal $P \leq 0.05$. SNP 6 is only marginally significantly associated unde the general genotypic model. Here is a testing script (no need to reproduce that, just check the results):

```
> for (i in 1:10) {
+ snpname <- paste("snp",i,sep="")
+ cat("\nTesting association between qt and SNP",snpname,":\n")
+ print(anova(lm(qt~get(snpname)),test="Chisq"))
+ #print(summary(lm(qt~get(snpname)))$coef)
+ }

Testing association between qt and SNP snp1 :
Analysis of Variance Table

Response: qt
              Df Sum Sq Mean Sq F value  Pr(>F)
get(snpname)   2    7.8  3.8995  3.9845 0.01873 *
Residuals   2371 2320.4  0.9787
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between qt and SNP snp2 :
Analysis of Variance Table

Response: qt
              Df  Sum Sq Mean Sq F value Pr(>F)
get(snpname)   1    0.18 0.18376  0.1841 0.6679
Residuals   2372 2367.23 0.99799
```

```
Testing association between qt and SNP snp3 :
Analysis of Variance Table

Response: qt
               Df  Sum Sq Mean Sq F value Pr(>F)
get(snpname)    2    3.24 1.61986   1.658 0.1907
Residuals    2375 2320.41 0.97701

Testing association between qt and SNP snp4 :
Analysis of Variance Table

Response: qt
               Df  Sum Sq Mean Sq F value  Pr(>F)
get(snpname)    2    7.68  3.8417  3.8628 0.02114 *
Residuals    2387 2373.94  0.9945
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between qt and SNP snp5 :
Analysis of Variance Table

Response: qt
               Df  Sum Sq Mean Sq F value  Pr(>F)
get(snpname)    2    6.48  3.2418  3.2798 0.03781 *
Residuals    2380 2352.48  0.9884
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between qt and SNP snp6 :
Analysis of Variance Table

Response: qt
               Df  Sum Sq Mean Sq F value  Pr(>F)
get(snpname)    2    5.49 2.74680  2.7808 0.06219 .
Residuals    2377 2347.91 0.98776
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between qt and SNP snp7 :
Analysis of Variance Table

Response: qt
               Df  Sum Sq Mean Sq F value Pr(>F)
get(snpname)    2    4.02 2.01212  2.0368 0.1307
Residuals    2365 2336.31 0.98787

Testing association between qt and SNP snp8 :
Analysis of Variance Table

Response: qt
```

```
               Df  Sum Sq Mean Sq F value Pr(>F)
get(snpname)    2    1.38 0.68987  0.6924 0.5005
Residuals    2368 2359.23 0.99630

Testing association between qt and SNP snp9 :
Analysis of Variance Table

Response: qt
               Df Sum Sq Mean Sq F value    Pr(>F)
get(snpname)    2   15.6  7.8014  7.9982 0.0003453 ***
Residuals    2358 2300.0  0.9754
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing association between qt and SNP snp10 :
Analysis of Variance Table

Response: qt
               Df  Sum Sq Mean Sq F value Pr(>F)
get(snpname)    2    1.19 0.59456  0.6041 0.5467
Residuals    2381 2343.47 0.98424
```

Answer (Ex. 16) — For 'snp1', though the data are compatible with either additive, dominant or recessive model, the additive model provides best fit to the data (largest p-value), while the recessive 'B' model provide the wors fit (almost significantly worse than the general model):

```
> table(snp1,as.numeric(snp1))
snp1     1    2    3
  A/A 1287    0    0
  A/B    0  888    0
  B/B    0    0  199
> table(snp1,(as.numeric(snp1)>=2))
snp1  FALSE TRUE
  A/A  1287    0
  A/B     0  888
  B/B     0  199
> table(snp1,(as.numeric(snp1)>=3))
snp1  FALSE TRUE
  A/A  1287    0
  A/B   888    0
  B/B     0  199
> model_gen <- lm(qt~snp1)
> summary(model_gen)
Call:
lm(formula = qt ~ snp1)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-3.5261 -0.6643 -0.0111  0.6765  3.5462

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
> model_add <- lm(qt~as.numeric(snp1))
> model_dom <- lm(qt~I(as.numeric(snp1)>=2))
> model_rec <- lm(qt~I(as.numeric(snp1)>=3))
> anova(model_add,model_gen,test="Chisq")
Analysis of Variance Table

Model 1: qt ~ as.numeric(snp1)
Model 2: qt ~ snp1
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2372 2320.5
2   2371 2320.4  1   0.04886   0.8232
> anova(model_dom,model_gen,test="Chisq")
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(snp1) >= 2)
Model 2: qt ~ snp1
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2372 2322.2
2   2371 2320.4  1    1.7733   0.1783
> anova(model_rec,model_gen,test="Chisq")
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(snp1) >= 3)
Model 2: qt ~ snp1
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2372 2324.0
2   2371 2320.4  1    3.5332  0.05743 .
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

For SNPs 4, 5, 6, and 9 results are:

```
> for (i in c(4,5,6,9)) {
+         snpname <- paste("snp",i,sep="")
```

```
+                 cat("\nTesting SNP",snpname,":\n")
+                 cursnp <- get(snpname)
+                 model_gen <- lm(qt~cursnp)
+                 print(summary(model_gen))
+                 model_add <- lm(qt~as.numeric(cursnp))
+                 model_dom <- lm(qt~I(as.numeric(cursnp)>=2))
+                 model_rec <- lm(qt~I(as.numeric(cursnp)>=3))
+                 print(anova(model_add,model_gen,test="Chisq"))
+                 print(anova(model_dom,model_gen,test="Chisq"))
+                 print(anova(model_rec,model_gen,test="Chisq"))
+         }

Testing SNP snp4 :

Call:
lm(formula = qt ~ cursnp)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4311 -0.6636 -0.0013  0.6737  3.5489

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.02132    0.02972   0.717   0.4733
cursnpA/A    0.02953    0.04423   0.668   0.5044
cursnpB/B   -0.14481    0.06192  -2.339   0.0194 *
---
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: 0.9973 on 2387 degrees of freedom
  (110 observations deleted due to missingness)
Multiple R-squared:  0.003226,        Adjusted R-squared:  0.002391
F-statistic: 3.863 on 2 and 2387 DF,  p-value: 0.02114

Analysis of Variance Table

Model 1: qt ~ as.numeric(cursnp)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2388 2375.7
2   2387 2373.9  1    1.7489   0.1848
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(cursnp) >= 2)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2388 2379.4
2   2387 2373.9  1    5.4391  0.01936 *
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

```
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(cursnp) >= 3)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2388 2374.4
2   2387 2373.9  1   0.44342   0.5043

Testing SNP snp5 :

Call:
lm(formula = qt ~ cursnp)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4719 -0.6589 -0.0084  0.6622  3.5285

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01401    0.02878   0.487   0.6264
cursnpA/A   -0.09667    0.05611  -1.723   0.0851 .
cursnpB/B    0.05727    0.04607   1.243   0.2140
---
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: 0.9942 on 2380 degrees of freedom
  (117 observations deleted due to missingness)
Multiple R-squared:  0.002749,        Adjusted R-squared:  0.00191
F-statistic:  3.28 on 2 and 2380 DF,  p-value: 0.03781

Analysis of Variance Table

Model 1: qt ~ as.numeric(cursnp)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2381 2352.7
2   2380 2352.5  1   0.22152   0.6359
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(cursnp) >= 2)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2381 2354.0
2   2380 2352.5  1    1.5273   0.2138
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(cursnp) >= 3)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2381 2355.4
```

```
2   2380 2352.5  1    2.9335  0.08494 .
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Testing SNP snp6 :

Call:
lm(formula = qt ~ cursnp)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4784 -0.6753 -0.0064  0.6703  3.5324

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.07617    0.05085  -1.498   0.1343
cursnpB/A    0.09417    0.05886   1.600   0.1097
cursnpB/B    0.14351    0.06096   2.354   0.0186 *
---
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: 0.9939 on 2377 degrees of freedom
  (120 observations deleted due to missingness)
Multiple R-squared:  0.002334,        Adjusted R-squared:  0.001495
F-statistic: 2.781 on 2 and 2377 DF,  p-value: 0.06219

Analysis of Variance Table

Model 1: qt ~ as.numeric(cursnp)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2378 2348.2
2   2377 2347.9  1   0.27462    0.598
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(cursnp) >= 2)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2378 2349.1
2   2377 2347.9  1    1.1967    0.271
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(cursnp) >= 3)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2378 2350.4
2   2377 2347.9  1    2.5284   0.1096

Testing SNP snp9 :
```

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

Residuals:
    Min      1Q  Median      3Q     Max
-3.5482 -0.6673  0.0074  0.6546  3.6061

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.006298   0.021562  -0.292  0.77026
cursnpA/B    0.162230   0.065729   2.468  0.01365 *
cursnpB/B    1.002439   0.313057   3.202  0.00138 **
---
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: 0.9876 on 2358 degrees of freedom
  (139 observations deleted due to missingness)
Multiple R-squared:  0.006738,        Adjusted R-squared:  0.005896
F-statistic: 7.998 on 2 and 2358 DF,  p-value: 0.0003453

Analysis of Variance Table

Model 1: qt ~ as.numeric(cursnp)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2359 2303.9
2   2358 2300.0  1    3.9528  0.04411 *
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(cursnp) >= 2)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2359 2306.8
2   2358 2300.0  1    6.7911 0.008324 **
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
Analysis of Variance Table

Model 1: qt ~ I(as.numeric(cursnp) >= 3)
Model 2: qt ~ cursnp
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1   2359 2305.9
2   2358 2300.0  1     5.942  0.01358 *
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

Chapter 4

Introduction to the GenABEL-package

In this section, you will become familiar with the GenABEL-package library, designed for GWA analysis. Compared to the 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 the GenABEL-package library using the command

```
> library(GenABEL)
```

After that, load the example data set using 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 R 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 (how are the alleles coded? – A, T, G, C? – as well as 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 to millions 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-package, a special data class, gwaa.data-class is used to store GWA data.

You may consider an object of gwaa.data-class as a 'black box' from which you can get specific data using specific functions. If you are interested in the internal structure of the gwaa.data-class, you can find the description in section B.1 (Internal structure of gwaa.data-class).

The data frame, which contains all phenotypic data in the study may be accessed using the `phdata` function. Let us have a look at the first few rows of the phenotypic data frame of `srdta`:

```
> phdata(srdta)[1:5,]
```

```
   id sex  age   qt1    qt2  qt3 bt
p1 p1   1 43.4 -0.58   4.46 1.43  0
p2 p2   1 48.2  0.80   6.32 3.90  1
p3 p3   0 37.9 -0.52   3.26 5.05  1
p4 p4   1 53.8 -1.55 888.00 3.76  1
p5 p5   1 47.5  0.25   5.70 2.89  1
```

The rows of this data frame correspond to th estudy 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, numer, or alphanumeric combination, 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, as its structure is coupled to the structure of genotypic data. If at some point you need to manipulate (add/delete) the phenotypes included in `phdata`, you need to use such `GenABEL-package` functions as `add.phdata` and `del.phdata` (see section 4.2).

The other part of an object of `gwaa.data-class` is `gtdata`, which contains all GWA genetic information in an object of class `snp.data` class. It is not supposed to be modified directly by user. The genotypic data can be accessed through the `gtdata` function, e.g.

```
> gtdata(srdta[1:10, 1:10])
```

```
@nids = 10
@nsnps = 10
@nbytes = 3
@idnames = p1 p2 p3 p4 p5 p6 p7 p8 p9 p10
@snpnames = rs10 rs18 rs29 rs65 rs73 rs114 rs128 rs130 rs143 rs150
@chromosome = 1 1 1 1 1 1 1 1 1 1
@coding =  08 0b 0c 03 04 03 0c 04 08 0f
@strand =  01 01 02 01 01 01 02 01 01 01
@map = 2500 3500 5750 13500 14250 24500 27000 27250 31000 33250
@male = 1 1 0 1 1 0 0 1 0 0
@gtps =
40 40 40 80 40 40 40 40 c0 c0
40 40 00 00 40 40 40 c0 40 40
40 40 00 80 40 40 40 40 c0 c0
```

As you can see, these data are of little direct use as these are stored in an internal format – you need to coerce that to another data type if you want to manipulate/analyse these data using non-`GenABEL-package` functions (see section **??**).

The number of individuals described in an object of `gwaa.data-class` can be accessed through `nids` function, e.g.

```
> nids(srdta)

[1] 2500
```

and the number of SNPs using the `nsnps` function:

```
> nsnps(srdta)

[1] 833
```

The IDs of the individuals included in the study can be accessed via the `idnames` function, for example the IDs of the first 7 individuals in the study are

```
> idnames(srdta)[1:7]

[1] "p1" "p2" "p3" "p4" "p5" "p6" "p7"
```

The sex of the individuals can be accessed using the `male` function:

```
> male(srdta)[1:7]

p1 p2 p3 p4 p5 p6 p7
 1  1  0  1  1  0  0
```

where males (heterogametic sex) are assigned with "1" and a homogametic sex (females) are assigned the value "0".

Names of SNPs can be accessed using the `snpnames` function; for example the names of the first 10 SNPs in the `srdta` are

```
> snpnames(srdta)[1:10]

 [1] "rs10"  "rs18"  "rs29"  "rs65"  "rs73"  "rs114" "rs128" "rs130" "rs143"
[10] "rs150"
```

SNP annotation includes (shown for the first 10 SNPs only):

- Chromosome:

```
> chromosome(srdta)[1:10]

 rs10  rs18  rs29  rs65  rs73 rs114 rs128 rs130 rs143 rs150
  "1"   "1"   "1"   "1"   "1"   "1"   "1"   "1"   "1"   "1"
```

- Map position

```
> map(srdta)[1:10]

 rs10  rs18  rs29  rs65  rs73 rs114 rs128 rs130 rs143 rs150
 2500  3500  5750 13500 14250 24500 27000 27250 31000 33250
```

- Coding (where the second allele is the "effect" or "coded" one):

```
> coding(srdta)[1:10]

 rs10  rs18  rs29  rs65  rs73 rs114 rs128 rs130 rs143 rs150
 "TG"  "GA"  "GT"  "AT"  "AG"  "AT"  "GT"  "AG"  "TG"  "CA"
```

For every SNP, the coding is represented with a pair of characters, for example "AG". For an "AG" polymorphism, you may expect "AA", "AG" and "GG" genotypes to be found in your 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. You can also access the reference allele with the method `refallele`

```
> refallele(srdta)[1:10]
```

```
 rs10  rs18  rs29  rs65  rs73 rs114 rs128 rs130 rs143 rs150
  "T"   "G"   "G"   "A"   "A"   "A"   "G"   "A"   "T"   "C"
```

and the effective (or 'coded') allelel with

```
> effallele(srdta)[1:10]
```

```
 rs10  rs18  rs29  rs65  rs73 rs114 rs128 rs130 rs143 rs150
  "G"   "A"   "T"   "T"   "G"   "T"   "T"   "G"   "G"   "A"
```

- The strand on which the coding is reported ('+', '-' or missing, 'u'):

```
> strand(srdta)[1:10]
```

```
 rs10  rs18  rs29  rs65  rs73 rs114 rs128 rs130 rs143 rs150
  "+"   "+"   "-"   "+"   "+"   "+"   "-"   "+"   "+"   "+"
```

Summary:

- `GenABEL-package` uses a special data class, `gwaa.data-class`, to store GWA data.
- To access the content of an object of `gwaa.data-class`, a number of functions is used

Exercise 1. Exploring IDs in `srdta`

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 2. Exploring SNPs in `srdta`

Explore the SNPs contained in `srdta` using the functions to access SNP names (`snpnames`) and map (`map`) location

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 Accessing and modifying phenotypic data

As was already mentioned, the object returned by `phdata` contains phenotypic data and is a conventional data frame, wich obligatory includes 'id' and 'sex' variables, and ordered an a way that it couples to the genotypic data.

Being a data frame, `phdata` may be accessed using the corresponding methods:

```
> phdata(srdta)[1:5, ]

   id sex  age   qt1    qt2  qt3 bt
p1 p1   1 43.4 -0.58   4.46 1.43  0
p2 p2   1 48.2  0.80   6.32 3.90  1
p3 p3   0 37.9 -0.52   3.26 5.05  1
p4 p4   1 53.8 -1.55 888.00 3.76  1
p5 p5   1 47.5  0.25   5.70 2.89  1

> class(phdata(srdta))

[1] "data.frame"

> phdata(srdta)[1:5, 2]

[1] 1 1 0 1 1

> phdata(srdta)[1:5, "sex"]

[1] 1 1 0 1 1

> phdata(srdta)$sex[1:5]

[1] 1 1 0 1 1
```

The modification of the phenotypic data is performed using special methods, because of specific restrictions on phenotypic data frames. There are two main functions which allow you to add (`add.phdata`) and delete (`del.phdata`) phenotypes from `phdata` part of an object of `gwaa.data-class`.

For example, if you want to add a variable (say, the square of age) computed from the "age" variable of `srdta`

```
> phdata(srdta)[1:5, ]

   id sex  age   qt1    qt2  qt3 bt
p1 p1   1 43.4 -0.58   4.46 1.43  0
p2 p2   1 48.2  0.80   6.32 3.90  1
p3 p3   0 37.9 -0.52   3.26 5.05  1
p4 p4   1 53.8 -1.55 888.00 3.76  1
p5 p5   1 47.5  0.25   5.70 2.89  1

> age2 <- phdata(srdta)$age^2
```

you need to use the `add.phdata` function:

```
> srdta <- add.phdata(srdta, newph=age2, name="age_squared")
> phdata(srdta)[1:5, ]

   id sex  age   qt1    qt2  qt3 bt age_squared
p1 p1   1 43.4 -0.58   4.46 1.43  0     1883.56
p2 p2   1 48.2  0.80   6.32 3.90  1     2323.24
p3 p3   0 37.9 -0.52   3.26 5.05  1     1436.41
p4 p4   1 53.8 -1.55 888.00 3.76  1     2894.44
p5 p5   1 47.5  0.25   5.70 2.89  1     2256.25
```

You can add more than one variable at once using the same function, however, in this case the second ("newph") argument of the function should be a data frame, which contains an 'id' variable specifing the IDs of the individuals. Imagine we have the data for individuals 'p1', 'p2' and 'p7' (we will generate random data for them; pay attention only to the result):

```
> newvalues <- matrix( rnorm(3*5), 3, 5 )
> newdata   <- data.frame(id=c("p1", "p2", "p7"),
+                         ph1=1, ph2=1, ph3=1, ph4=1, ph5=1)
> newdata[, c(2:6)] <- newvalues
> newdata

  id        ph1         ph2        ph3       ph4        ph5
1 p1 -0.7456830  0.07034531  0.6118969  1.017129 -0.2248321
2 p2  1.1205270 -1.43527067  0.4614292 -1.016423  2.0098503
3 p7 -0.9747375  0.18167733 -1.5388997  1.345695 -0.2691746
```

These data can be added to the phenotypic data with

```
> srdta <- add.phdata(srdta,newdata)
> phdata(srdta)[1:10, ]

     id sex  age   qt1    qt2  qt3 bt age_squared        ph1         ph2
p1   p1   1 43.4 -0.58   4.46 1.43  0     1883.56 -0.7456830  0.07034531
p2   p2   1 48.2  0.80   6.32 3.90  1     2323.24  1.1205270 -1.43527067
p3   p3   0 37.9 -0.52   3.26 5.05  1     1436.41         NA          NA
p4   p4   1 53.8 -1.55 888.00 3.76  1     2894.44         NA          NA
p5   p5   1 47.5  0.25   5.70 2.89  1     2256.25         NA          NA
p6   p6   0 45.0  0.15   4.65 1.87  0     2025.00         NA          NA
p7   p7   0 52.0 -0.56   4.64 2.49  0     2704.00 -0.9747375  0.18167733
p8   p8   1 42.5    NA   5.77 2.68  1     1806.25         NA          NA
p9   p9   0 29.7 -2.26   0.71 1.45  0      882.09         NA          NA
p10 p10   0 45.8 -1.32   3.26 0.85  0     2097.64         NA          NA
           ph3       ph4        ph5
p1   0.6118969  1.017129 -0.2248321
p2   0.4614292 -1.016423  2.0098503
p3          NA        NA         NA
p4          NA        NA         NA
p5          NA        NA         NA
p6          NA        NA         NA
p7  -1.5388997  1.345695 -0.2691746
p8          NA        NA         NA
```

```
p9          NA         NA          NA
p10         NA         NA          NA
```

Finally, if you need, you can delete some phenotypes from the `phdata` using `del.phdata` function. Let us delete the phenotypes we have just added:

```
> srdta <- del.phdata(srdta,
+                     c("age_squared", "ph1", "ph2", "ph3", "ph4", "ph5"))
> phdata(srdta)[1:10, ]
```

```
     id sex  age   qt1    qt2  qt3 bt
p1   p1   1 43.4 -0.58   4.46 1.43  0
p2   p2   1 48.2  0.80   6.32 3.90  1
p3   p3   0 37.9 -0.52   3.26 5.05  1
p4   p4   1 53.8 -1.55 888.00 3.76  1
p5   p5   1 47.5  0.25   5.70 2.89  1
p6   p6   0 45.0  0.15   4.65 1.87  0
p7   p7   0 52.0 -0.56   4.64 2.49  0
p8   p8   1 42.5    NA   5.77 2.68  1
p9   p9   0 29.7 -2.26   0.71 1.45  0
p10 p10   0 45.8 -1.32   3.26 0.85  0
```

Summary:

- Phenotypic data contained in an object of `gwaa.data-class` can be accessed using the `phdata` functions
- You can add phenotypes using the `add.phdata` function
- You can delete phenotypes using the `del.phdata` function

4.3 Sub-setting and coercing gwaa.data

It is possible to sub-set the object, which stores the GWA data in the manner similar to that used for conventional `R` matrices and data frames. 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 more complicated). For example, if we would like to investigate 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"
```

As you can see, by sub-setting we obtained a smaller object of `gwaa.data-class`. The two major parts it contains are phenotypic data, which can be accessed through `phdata` (discussed in section 4.2):

```
> phdata(ssubs)
```

```
   id sex  age   qt1    qt2  qt3 bt
p1 p1   1 43.4 -0.58   4.46 1.43  0
p2 p2   1 48.2  0.80   6.32 3.90  1
p3 p3   0 37.9 -0.52   3.26 5.05  1
p4 p4   1 53.8 -1.55 888.00 3.76  1
p5 p5   1 47.5  0.25   5.70 2.89  1
```

and genotypc data, which can be accessed via the `gtdata` function:

```
> gtdata(ssubs)
```

```
@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
```

whose content is not quite straightforward to read.

To get human-readable information, a genotypic object should be coerced to a regular `R` data type, e.g. character, using the `as.character()` function:

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

```
   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"
```

Another useful coercion is to ”numeric”:

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

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

Note that conversion to numeric happened according to the underlying genotype and the rules specified by SNP coding:

```
> coding(ssubs)

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

– the genotype, which is made of the 'first' allele of the 'code' is converted to "0", the heterozygote to "1" and a homozygote for the second allele is converted to "2".

For example, when the coding is "GA", as it 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` format can be used in any further analysis.

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 their own genetic data formats.

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

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

NOTE: THIS PACKAGE IS NOW OBSOLETE.

  The R-Genetics project has developed an set of enhanced genetics
  packages to replace 'genetics'. Please visit the project homepage
  at http://rgenetics.org for informtion.

   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-package` data to the format used by "haplo.stats" you can use function `as.hsgeno()`

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

   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-package provides a functional interface to "haplo.stats" (such GenABEL-package 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

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

         id sex  age   qt1  qt2  qt3 bt
p141   p141   0 47.2  0.51 5.23 2.17  0
p147   p147   0 43.2  0.14 4.47 1.73  0
p2000 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 a 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

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

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

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

@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 3. Exploring rs114

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** ("effect") allele in total sample?
3. What is the frequency of the effect allele in males?
4. What is the frequency of the effect allele in females?
5. What is the frequency of the **reference** allele in the total sample, males and females?

Summary:

- It is possible to obtain subsets of objects of `gwaa.data-class` and `snp.data-class` using the 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`.
- The function `as.numeric()` converts genotypic data from `snp.data-class` to regular integer numbers, which can be used in analysis with R.
- The function `as.character()` converts genotypic data from `snp.data-class` to character format.
- The function `as.genotype()` converts genotypic data from `snp.data-class` to the format used by the `genetics` library.
- The function `as.hsgeno()` converts genotypic data from `snp.data-class` to the format used by the `haplo.stats` library.

4.4 Exploring genetic data

Implementation of the function `summary()` to summarize the genotypic part of a `gwaa.data-class` object 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

     Chromosome Position Strand A1 A2 NoMeasured CallRate Q.2 P.11 P.12 P.22
rs10          1     2500      +  T  G          5      1.0 0.0    5    0    0
rs18          1     3500      +  G  A          5      1.0 0.1    4    1    0
rs29          1     5750      -  G  T          2      0.4 0.0    2    0    0
     Pexact       Fmax      Plrt
rs10      1  0.0000000 1.0000000
rs18      1 -0.1111111 0.7386227
rs29      1  0.0000000 1.0000000
```

In the first section, a summary is generated for the phenotypic data. In the second section, a summary is generated for the 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 have seen above, an object of the class `gwaa.data-class` is subsettable in the standard manner: `[i, j]`, where `i` is the index of a study subject and `j` is the index of a SNP. Importantly, `i` could be a list of indexes:

```
> vec <- which(phdata(srdta)$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( gtdata(srdta[vec, 1:3]) )

     Chromosome Position Strand A1 A2 NoMeasured CallRate       Q.2 P.11 P.12
rs10          1     2500      +  T  G         48     0.96 0.1354167   36   11
rs18          1     3500      +  G  A         47     0.94 0.2765957   25   18
rs29          1     5750      -  G  T         45     0.90 0.1555556   32   12
     P.22    Pexact        Fmax      Plrt
rs10    1 1.0000000  0.02131603 0.8843626
rs18    4 0.7245853  0.04298643 0.7697067
rs29    1 1.0000000 -0.01503759 0.9188943
```

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

vec
FALSE  TRUE
 2450    50

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

```
     Chromosome Position Strand A1 A2 NoMeasured CallRate       Q.2 P.11 P.12
rs10          1     2500      +  T  G         48     0.96 0.1354167   36   11
rs18          1     3500      +  G  A         47     0.94 0.2765957   25   18
rs29          1     5750      -  G  T         45     0.90 0.1555556   32   12
     P.22    Pexact        Fmax      Plrt
rs10    1 1.0000000  0.02131603 0.8843626
rs18    4 0.7245853  0.04298643 0.7697067
rs29    1 1.0000000 -0.01503759 0.9188943
```

or a list with IDs of study subjects:

```
> vec1 <- idnames(srdta)[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( gtdata(srdta[vec1, 1:3]) )

     Chromosome Position Strand A1 A2 NoMeasured CallRate       Q.2 P.11 P.12
rs10          1     2500      +  T  G         48     0.96 0.1354167   36   11
rs18          1     3500      +  G  A         47     0.94 0.2765957   25   18
rs29          1     5750      -  G  T         45     0.90 0.1555556   32   12
     P.22    Pexact        Fmax      Plrt
rs10    1 1.0000000  0.02131603 0.8843626
rs18    4 0.7245853  0.04298643 0.7697067
rs29    1 1.0000000 -0.01503759 0.9188943
```

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

```
> a <- summary( gtdata(srdta[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 14

> names(a)

 [1] "Chromosome" "Position"   "Strand"     "A1"         "A2"
 [6] "NoMeasured" "CallRate"   "Q.2"        "P.11"       "P.12"
[11] "P.22"       "Pexact"     "Fmax"       "Plrt"
```

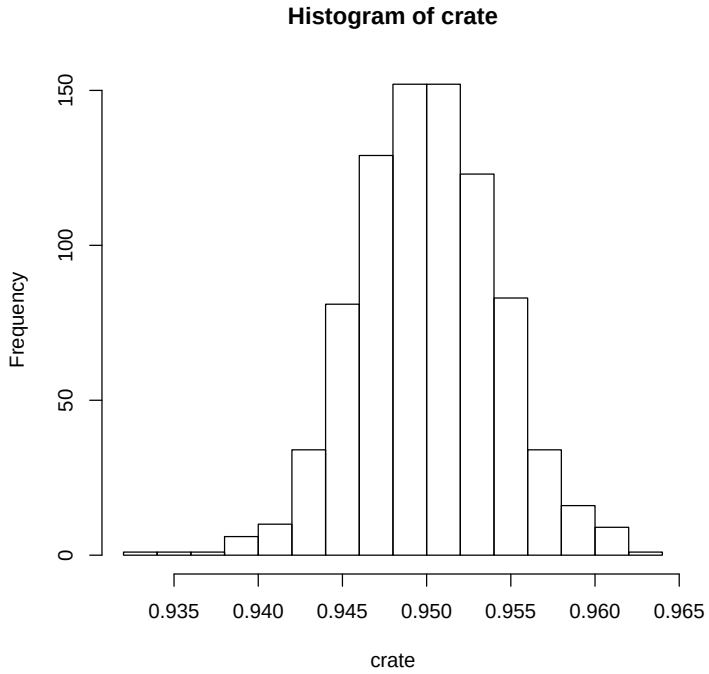

Figure 4.1: Histogram of the call rate.

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

Exercise 4. Testing HWE for 10 SNPs

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

This vector may be depicted by a histogram

```
> hist(crate)
```

which shows that most SNPs have call rate between 93 and 97% (Figure 4.1).

As a 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 the `catable()` command for that:

```
> catable(crate, c(.93, .95, .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
```

A similar procedure may be applied to see deviation from HWE:

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

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

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

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

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

     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 the 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 in Figure 4.2) and then generate a table describing the 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=TRUE)
```

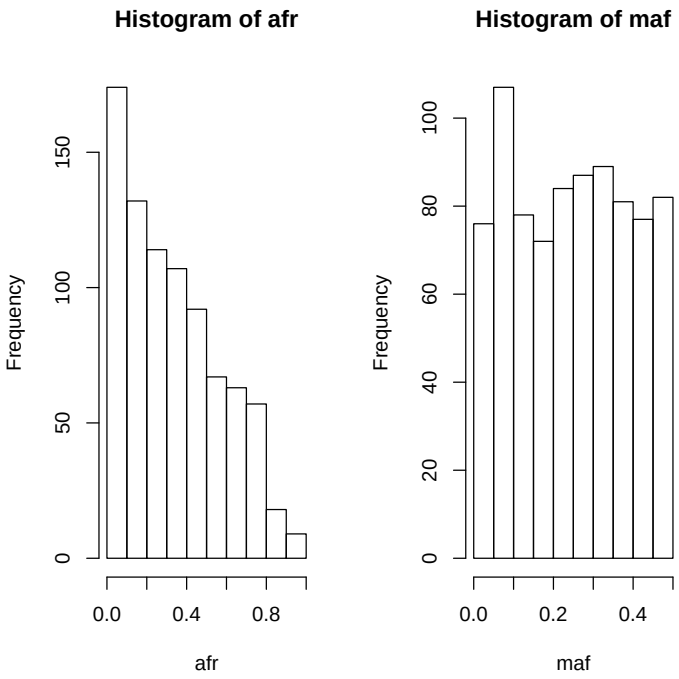

Figure 4.2: Histogram of the call rate.

```
     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 monomorphic SNPs which we recommend to exclude from analysis.

Another 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 NoPoly       Hom    E(Hom)       Var            F    CallPP
p1         790    707 0.7987342 0.6600319 0.4048662  0.407986159 0.9483794
p2         792    714 0.7474747 0.6585152 0.5090002  0.260508049 0.9507803
p3         783    700 0.6206897 0.6618209 0.4332890 -0.121625581 0.9399760
p4         789    705 0.6070976 0.6601276 0.5251900 -0.156029161 0.9471789
p5         790    707 0.6658228 0.6619821 0.5288936  0.011362319 0.9483794
p6         787    703 0.7662008 0.6622227 0.3770418  0.307830275 0.9447779
p7         794    709 0.6309824 0.6587669 0.4527349 -0.081423884 0.9531813
p8         793    711 0.7023960 0.6587232 0.5163296  0.127968868 0.9519808
p9         788    711 0.6675127 0.6573272 0.5599395  0.029723748 0.9459784
p10        797    713 0.6587202 0.6614644 0.4889042 -0.008105999 0.9567827
          Het
p1  0.2012658
p2  0.2525253
p3  0.3793103
p4  0.3929024
p5  0.3341772
p6  0.2337992
p7  0.3690176
p8  0.2976040
p9  0.3324873
p10 0.3412798
```

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

Let us analyse the distribution of heterozygosity:

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

[1] 0.3309457

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

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

> hist(het)
```

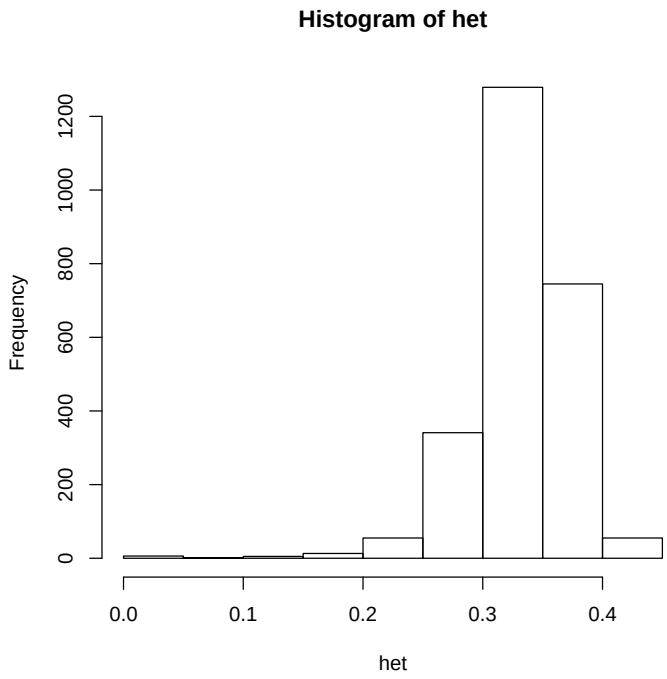

Figure 4.3: Histogram of heterozygosity.

The resulting histogram is shown in Figure 4.3. It is easy to see that a few people have very low heterozygosity, but there are no outliers with extremely high values.

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

Summary:

- When the `summary()` function is applied to a `gtdata` subset of `gwaa.data-class`, it returns summary statistics for the SNPs, including an exact test for Hardy-Weinberg equilibrium.
- When the `perid.summary()` function is applied to an object of `gwaa.data-class` (or the `gtdata` part of it), it returns per-person summary statistics, including the call rate within this person and its heterozygosity.

Exercise 5. Characterizing call rate

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

4.5 Answers to exercises

Answer (Ex. 1) — Load the data with

```
> data(srdta)
```

Number of people:

```
> nids(srdta)
```

```
[1] 2500
```

Number of males:

```
> sum(male(srdta))
```

```
[1] 1275
```

Number of females:

```
> nids(srdta) - sum(male(srdta))
```

```
[1] 1225
```

... you could get the same answer like this[1]:

```
> sum(male(srdta) == 0)
```

```
[1] 1225
```

[1] This is something covered later in the section 4.3 ("Sub-setting and coercing gwaa.data")

The proportion of males can be computed using above results

```
> sum(male(srdta)) / nids(srdta)
```

```
[1] 0.51
```

or by using the `mean()` function:

```
> mean(male(srdta))
```

```
[1] 0.51
```

Answer (Ex. 2) — The names of markers located after 2,490,000 b.p. are

```
> vec <- (map(srdta) > 2490000)
> snpnames(srdta)[vec]
```

```
[1] "rs9273" "rs9277" "rs9279" "rs9283"
```

The names of markers located between 1,100,000 and 1,105,000 b.p. are:

```
> vec <- (map(srdta) > 1100000 & map(srdta) < 1105000)
> snpnames(srdta)[vec]
```

```
[1] "rs4180" "rs4186" "rs4187"
```

Answer (Ex. 3) — To learn what allele of "rs114" is the reference you need to run

```
> coding(srdta)["rs114"]
```

```
rs114
 "AT"
```

Here, the first ("A") allele is the reference and thus the second ("T") is the effect allele. Remember that when using the `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 ("effect") homozygotes will be coded as "2":

```
> table(as.character( gtdata(srdta[, "rs114"] )),
+       as.numeric(   gtdata(srdta[,"rs114"] )))
```

```
         0    1    2
  A/A 1868    0    0
  A/T    0  491    0
  T/T    0    0   34
```

To compute the frequency of the effect allele of SNP "rs114" in the 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(gtdata(srdta[, "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=TRUE)
```

```
[1] 559
```

so the number of "effect" alleles is 559.
However, now we do not know what was the number of people for whom the genotype was measured! – `nids` would return the *total* number of people, but not the number of those measured for "rs114".
This problem can be dealt with through using the `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

```
> nids(srdta)

[1] 2500

> nmeasured <- sum(!is.na(a))
> nmeasured

[1] 2393
```

(note the "!" before `is.na`, which means NOT, so we get these elements which are not `NA`). Consequently, the frequency of the "effect" allele is

```
> sum(a, na.rm=TRUE) / (2 * nmeasured)

[1] 0.116799
```

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=TRUE)/2

[1] 0.116799
```

To compute the frequency of the effect allele of "rs114" in males, you can use

```
> amale <- as.numeric( gtdata(srdta[male(srdta)==1, "rs114"]) )
> mean(amale, na.rm=TRUE)/2

[1] 0.1164216
```

To compute the frequency of the effect allele in females, you can use

```
> afemale <- as.numeric( gtdata(srdta[male(srdta)==0, "rs114"]) )
> mean(afemale, na.rm=TRUE)/2

[1] 0.1171942
```

The frequencies of the reference allele are computed very simply as one minus the frequency of the effective allele:

```
> 1 - mean(a,       na.rm=TRUE)/2

[1] 0.883201

> 1 - mean(amale,   na.rm=TRUE)/2

[1] 0.8835784

> 1 - mean(afemale, na.rm=TRUE)/2

[1] 0.8828058
```

Answer (Ex. 4) — To test for HWE in first 10 SNPs in total sample

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

```
      Chromosome Position Strand A1 A2 NoMeasured CallRate        Q.2 P.11 P.12
rs10           1     2500      +  T  G       2384   0.9536 0.13255034 1792  552
rs18           1     3500      +  G  A       2385   0.9540 0.28029350 1232  969
rs29           1     5750      -  G  T       2374   0.9496 0.13774221 1763  568
rs65           1    13500      +  A  T       2378   0.9512 0.71972246  182  969
rs73           1    14250      +  A  G       2385   0.9540 0.01341719 2331   44
rs114          1    24500      +  A  T       2393   0.9572 0.11679900 1868  491
rs128          1    27000      -  G  T       2391   0.9564 0.02488499 2281  101
rs130          1    27250      +  A  G       2379   0.9516 0.69377890  222 1013
rs143          1    31000      +  T  G       2377   0.9508 0.47728229  655 1175
rs150          1    33250      +  C  A       2369   0.9476 0.65998312  267 1077
      P.22       Pexact         Fmax         Plrt
rs10    40 7.897327e-01 -0.006880004 7.355343e-01
rs18   184 7.608230e-01 -0.007017332 7.315304e-01
rs29    43 7.955141e-01 -0.007241148 7.227853e-01
rs65  1227 6.475412e-01 -0.010016746 6.246577e-01
rs73    10 1.792470e-12  0.303150234 1.281239e-12
rs114   34 7.663683e-01  0.005487764 7.894076e-01
rs128    9 9.408599e-06  0.129600629 1.000431e-05
rs130 1144 9.615127e-01 -0.002140946 9.168114e-01
rs143  547 6.512540e-01  0.009313705 6.497695e-01
rs150 1025 5.518478e-01 -0.012948436 5.281254e-01
```

To test it in cases

```
> summary( gtdata(srdta[phdata(srdta)$bt==1, 1:10]) )
```

```
      Chromosome Position Strand A1 A2 NoMeasured  CallRate        Q.2 P.11
rs10           1     2500      +  T  G       1197 0.9622186 0.13700919  888
rs18           1     3500      +  G  A       1189 0.9557878 0.28511354  605
rs29           1     5750      -  G  T       1176 0.9453376 0.14285714  859
rs65           1    13500      +  A  T       1185 0.9525723 0.72700422   83
rs73           1    14250      +  A  G       1187 0.9541801 0.01053075 1167
rs114          1    24500      +  A  T       1190 0.9565916 0.12184874  918
rs128          1    27000      -  G  T       1183 0.9509646 0.02409129 1129
rs130          1    27250      +  A  G       1188 0.9549839 0.68392256  117
rs143          1    31000      +  T  G       1192 0.9581994 0.48489933  320
rs150          1    33250      +  C  A       1182 0.9501608 0.66624365  127
      P.12 P.22       Pexact         Fmax         Plrt
rs10   290   19 4.635677e-01 -0.024514202 3.871421e-01
rs18   490   94 7.759191e-01 -0.010949158 7.052930e-01
rs29   298   19 2.832575e-01 -0.034722222 2.214580e-01
rs65   481  621 4.647357e-01 -0.022595469 4.348023e-01
rs73    15    5 3.988770e-08  0.393614304 2.423624e-08
rs114  254   18 8.924018e-01  0.002606831 9.285104e-01
rs128   51    3 2.747904e-02  0.083175674 3.157174e-02
rs130  517  554 8.407527e-01 -0.006569292 8.207476e-01
rs143  588  284 6.848365e-01  0.012522119 6.654994e-01
rs150  535  520 5.568363e-01 -0.017756050 5.409408e-01
```

in controls

```
> summary( gtdata(srdta[phdata(srdta)$bt==0, 1:10]) )
```

```
      Chromosome Position Strand A1 A2 NoMeasured  CallRate        Q.2 P.11
rs10           1     2500      +  T  G       1177 0.9453815 0.12744265  897
rs18           1     3500      +  G  A       1185 0.9518072 0.27426160  623
rs29           1     5750      -  G  T       1188 0.9542169 0.13215488  897
rs65           1    13500      +  A  T       1183 0.9502008 0.71344041   98
rs73           1    14250      +  A  G       1188 0.9542169 0.01641414 1154
rs114          1    24500      +  A  T       1192 0.9574297 0.11157718  941
rs128          1    27000      -  G  T       1197 0.9614458 0.02589808 1141
rs130          1    27250      +  A  G       1181 0.9485944 0.70491109  104
rs143          1    31000      +  T  G       1174 0.9429719 0.46805792  334
rs150          1    33250      +  C  A       1176 0.9445783 0.65306122  139
      P.12 P.22       Pexact         Fmax         Plrt
rs10   260   20 7.933317e-01  0.006751055 8.178295e-01
rs18   474   88 9.418133e-01 -0.004812165 8.683219e-01
rs29   268   23 5.288436e-01  0.016525913 5.737373e-01
rs65   482  603 8.871139e-01  0.003540522 9.031273e-01
rs73    29    5 6.941219e-06  0.244001185 5.537568e-06
rs114  236   15 8.846527e-01  0.001356081 9.627084e-01
rs128   50    6 7.745807e-05  0.172107564 7.552399e-05
rs130  489  588 8.887439e-01  0.004728114 8.710047e-01
rs143  581  259 8.604122e-01  0.006165442 8.326938e-01
rs150  538  499 7.968462e-01 -0.009574142 7.424986e-01
```

SNPs 'rs73' and 'rs128' are out of HWE (at $p \leq 0.05$) in the total sample, and also in cases and controls.

Answer (Ex. 5) — To characterize ID call rate, you can run the following commands:

```
> idsummary <- perid.summary(srdta)
> idsummary[1:5, ]
   NoMeasured NoPoly       Hom    E(Hom)       Var           F    CallPP
p1        790    790 0.7987342 0.6696986 0.5448255  0.3906601 0.9483794
p2        792    792 0.7474747 0.6685502 0.5390602  0.2381191 0.9507803
p3        783    783 0.6206897 0.6712102 0.4888671 -0.1536561 0.9399760
p4        789    789 0.6070976 0.6700900 0.4077382 -0.1909382 0.9471789
p5        790    790 0.6658228 0.6710232 0.4340010 -0.0158077 0.9483794
         Het
p1 0.2012658
p2 0.2525253
p3 0.3793103
p4 0.3929024
p5 0.3341772
> idcall <- idsummary$Call
> idcall[1:5]
[1] 0.9483794 0.9507803 0.9399760 0.9471789 0.9483794
> catable(idcall, c(0.9, 0.93, 0.95, 0.98, 0.99))
     X<=0.9 0.9<X<=0.93 0.93<X<=0.95 0.95<X<=0.98 0.98<X<=0.99 X>0.99
No        0      13.000     1186.000      1301.00            0      0
Prop      0       0.005        0.474         0.52            0      0
```

```
>   table(idcall < 0.93)
FALSE  TRUE
 2487    13
```

To produce a histogram of call rates, use `hist(idcall)`

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 really associated ones) are specific features of this data set, which are not observed in true GWA studies.

Start R and load the `GenABEL-package` library by typing

```
> library(GenABEL)
```

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

```
> data(ge03d2ex)
```

Investigate the objects loaded by the command

```
> ls()
```

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

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(phdata(ge03d2ex))

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

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

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

5.1 Data descriptives and first round of GWA analysis

Let us investigate which traits are present in the loaded data frame and what are the characteristics of the distribution by using the GenABEL-package 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 the phenotypic frame contains 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 proportion of successful genotypes (per person)`
     X<=0.9 0.9<X<=0.95 0.95<X<=0.98 0.98<X<=0.99 X>0.99
No    1.000           0            0      135.000      0
Prop  0.007           0            0        0.993      0

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

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

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

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

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

It is of note that we can see inflation of the proportion of the tests for HWE at a 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 proportion of successful genotypes (per person)`
     X<=0.9 0.9<X<=0.95 0.95<X<=0.98 0.98<X<=0.99 X>0.99
No        0           0            0           50      0
Prop      0           0            0            1      0

$`Distribution of proportion 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] 0.252572

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

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

Let us check whether there are indications that deviation from HWE is due to cases. At this stage we are only interested in the HWE distribution table, and therefore will ask to report the distrbution for cases (`dm2==1`) and 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
```

and for the controls

```
> descriptives.marker(ge03d2ex, ids=(dm2==0))[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          0    3.000  14.000  98.000  4000
Prop        0    0.001   0.003   0.024     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 genetic heterogeneity specific for cases.

In essence, the '`descriptives.marker`' function uses the '`summary`' function to generate the HW P-values distribution. It may be interesting to generate this distribution using the '`summary`' function You do not need to do so, but this example shows how you can generate summaries from underlying SNP-tables. First, we need to compute summary SNP statistics by

```
> s <- summary( gtdata( ge03d2ex[(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 the first 5 rows of this very long summary table by

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

```
          Chromosome Position Strand A1 A2 NoMeasured  CallRate        Q.2 P.11
rs1646456          1      653      +  C  G         85 0.9883721 0.32941176   34
rs4435802          1     5291      +  C  A         86 1.0000000 0.09302326   70
rs946364           1     8533      -  T  C         84 0.9767442 0.24404762   46
rs299251           1    10737      +  A  G         85 0.9883721 0.03529412   79
rs2456488          1    11779      +  G  C         85 0.9883721 0.33529412   38
          P.12 P.22     Pexact         Fmax       Plrt
rs1646456   46    5 0.05089075 -0.22493734 0.03233812
rs4435802   16    0 1.00000000 -0.10256410 0.19978917
rs946364    35    3 0.37413083 -0.12924909 0.21777250
rs299251     6    0 1.00000000 -0.03658537 0.63937447
rs2456488   37   10 0.81059718  0.02344356 0.82918345
```

Note that the column 'Pexact' provides exact HWE test P-values we need. We can extract these to a separate vector by

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

and do characterization of the cummulative distribution by

```
> catable(pexcas, c(0.001,0.01,0.05,0.1,0.5), cumulative=TRUE)
```

```
     X<=0.001 X<=0.01 X<=0.05  X<=0.1  X<=0.5 all X
No      79.00 136.000 268.000 390.000 1359.00  4000
Prop     0.02   0.034   0.067   0.098    0.34     1
```

You can generate the distribution for controls in similar manner.

Let us first try do a GWA scan using the raw (before quality control) data. We will use the score test, as implemented in the `qtscore()`[1] function of `GenABEL-package` for testing:

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

[1] consider '`mlreg`' function if you want to run a true linear regression

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 information is computed by this function by using

```
> an0
```

```
***** 'scan.gwaa' object *****
*** Produced with:
qtscore(formula = dm2, data = ge03d2ex, trait.type = "binomial")
*** Test used: binomial
*** no. IDs used: 136 ( id199 id287 id300 , ... )
*** Lambda: 1.033102
*** Results table contains 4000 rows and 9 columns
*** Output for 10 first rows is:
            N      effB    se_effB     chi2.1df      P1df     effAB     effBB
rs1646456 135 0.9487666  5.0501959 0.0352941176 0.8509807 1.5558824 0.4831933
rs4435802 134 2.6822601  1.6765631 2.5595403237 0.1096305 2.5142857        NA
rs946364  134 0.6376645  0.4012104 2.5260391714 0.1119810 0.7277883 0.2869565
rs299251  135 0.5592122  0.5740215 0.9490674319 0.3299568 0.5569620        NA
rs2456488 135 0.8669860  1.5393112 0.3172278551 0.5732784 0.9736842 0.6907895
rs3712159 133 0.8282737  2.4803134 0.1115153279 0.7384255 0.5641026       Inf
rs4602970 136 1.5227297  2.2683336 0.4506421219 0.5020302 1.5131579        NA
rs175910  134 0.9949826 57.3970087 0.0003005055 0.9861693 0.5600000 4.0727273
rs1919938 136 0.9303079  3.5515057 0.0686164762 0.7933619 0.1160000 0.1958333
rs8892781 133 1.0953022 14.9055712 0.0053997133 0.9414220 1.0952381        NA
             chi2.2df       P2df
rs1646456 3.885667042 0.14329734
rs4435802 2.559540324 0.10963046
rs946364  3.020970009 0.22080286
rs299251  0.949067432 0.32995679
rs2456488 0.493411146 0.78137072
rs3712159 1.358996877 0.50687116
rs4602970 0.450642122 0.50203018
rs175910  5.012993241 0.08155345
rs1919938 6.001763944 0.04974318
rs8892781 0.005399713 0.94142198
   ...
___ Use 'results(object)' to get complete results table ___
```

Here, let us look at the 'Results table'. `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 in the `lambda` object). `effB` corresponds to the (approximate) Odds Ratio estimate for the SNP.

Let us see if there is evidence for the inflation of the test statistics; for that let us obtain λ with

```
> lambda(an0)
```

```
$estimate
[1] 1.033102
```

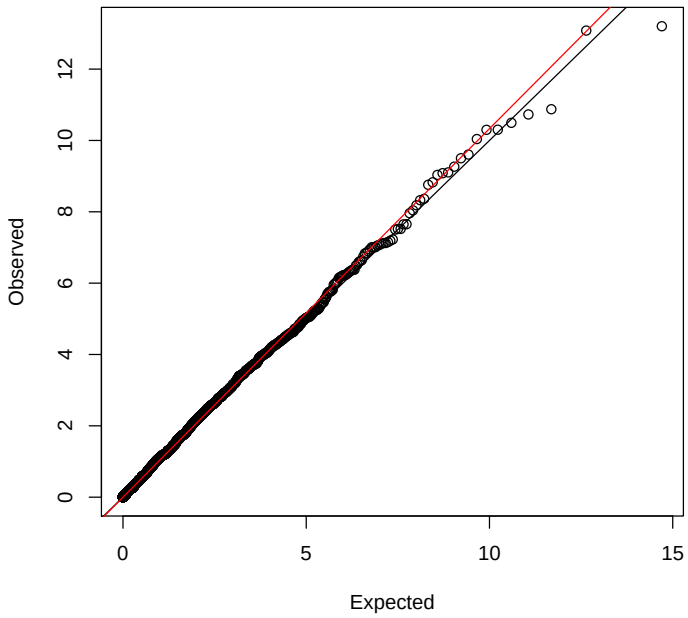

Figure 5.1: $\chi^2 - \chi^2$ plot for a GWA scan. Black line of slope 1: expected under no inflation; Red line: fitted slope.

```
$se
[1] 0.0005639231
```

The estimate of λ is 1.03, suggesting inflation of the test and some degree of stratification. Though the value obtained seems to be small, it should be noted that λ grows linearly with sample size, so for this small number of cases and controls the value is worrisome.

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; this was already done automatically when running `qtscore` function, but let us repeat this manually:

```
> estlambda(an0[, "P1df"], plot=TRUE)

$estimate
[1] 1.033102

$se
[1] 0.0005639231
```

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

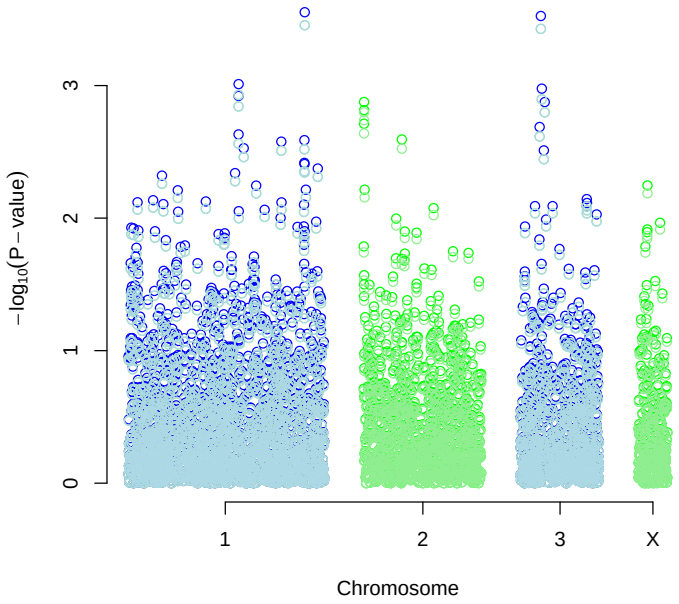

Figure 5.2: $-\log_{10}(P\text{-value})$ from the genome scan before QC procedure. Raw analysis: darker circles; corrected analysis: lighter circles

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

We can also present the obtained results using the "Manhatten plot", where the SNP genomic position is on the horizontal axis and $-\log_{10}$ of the P-value is shown on the vertical axis:

```
> plot(an0)
```

The resulting plot is show in Figure 5.2. By default, $-\log_{10}(P\text{-value})$ of the uncorrected 1 d.f. test are shown; see thehelp 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 see that the P-values corrected by genomic control are uniformly lower than the P-values from 'raw' analysis. This is to be expected as genomic control simply divides the 'raw' χ^2 statistics by a constant λ for all SNPs.

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

```
> descriptives.scan(an0)

Summary for top 10 results, sorted by P1df
          Chromosome Position Strand A1 A2   N       effB    se_effB  chi2.1df
rs1719133          1  4495479      +  T  A 136 0.33729339 0.09282784 13.202591
rs2975760          3 10518480      +  A  T 134 3.80380024 1.05172986 13.080580
rs7418878          1  2808520      +  A  T 136 3.08123060 0.93431795 10.875745
rs5308595          3 10543128      -  C  G 133 3.98254950 1.21582875 10.729452
rs4804634          1  2807417      +  C  G 132 0.43411456 0.13400290 10.494949
rs3224311          2  6009769      +  G  C 135 3.15831710 0.98401491 10.301681
rs26325            3 10617781      +  A  C 135 0.09742793 0.03035964 10.298495
rs8835506          2  6010852      +  A  T 132 3.17720829 1.00274087 10.039543
rs3925525          2  6008501      +  C  G 135 2.98416931 0.96286458  9.605423
rs2521089          3 10487652      -  T  C 135 2.50239493 0.81179595  9.502064
                  P1df     effAB     effBB  chi2.2df         P2df        Pc1df
rs1719133 0.0002795623 0.4004237  0.000000 14.729116 0.0006333052 0.0003504258
rs2975760 0.0002983731 3.4545455 10.000000 13.547345 0.0011434877 0.0003732694
rs7418878 0.0009743183 3.6051282  4.871795 12.181064 0.0022642036 0.0011762545
rs5308595 0.0010544366 3.3171429       Inf 10.766439 0.0045930101 0.0012699705
rs4804634 0.0011970132 0.5240642  0.173913 11.200767 0.0036964462 0.0014362332
rs3224311 0.0013290907 3.4151786  4.250000 11.658283 0.0029405999 0.0015897278
rs26325   0.0013313876 0.1097724        NA 10.298495 0.0013313876 0.0015923930
rs8835506 0.0015321522 3.4903846  4.125000 11.513206 0.0031618340 0.0018248521
rs3925525 0.0019400358 3.2380952  4.121212 10.782867 0.0045554384 0.0022944719
rs2521089 0.0020524092 2.5717703  4.772727  9.933387 0.0069661425 0.0024233145
```

or, equivalently, by 'summary(an0)'

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 than 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 the three possible genotypes were observed, and consequently the 2 d.f. test could not be performed.

Now let us apply the 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)

  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

(you may skip the 'quiet=TRUE' argument, then you will see progress)

Now let us generate the summary of the results

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

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2   N       effB    se_effB  chi2.1df
rs1719133          1  4495479      +  T  A 136 -0.2652064 0.07298850 13.202591
```

```
rs2975760          3 10518480      +  A  T 134  0.2340655 0.06471782 13.080580
rs7418878          1  2808520      +  A  T 136  0.2089098 0.06334746 10.875745
rs5308595          3 10543128      -  C  G 133  0.2445516 0.07465893 10.729452
rs4804634          1  2807417      +  C  G 132 -0.2050449 0.06329344 10.494949
rs3224311          2  6009769      +  G  C 135  0.2133633 0.06647611 10.301681
rs26325            3 10617781      +  A  C 135 -0.4875367 0.15192190 10.298495
rs8835506          2  6010852      +  A  T 132  0.2112000 0.06665565 10.039543
rs3925525          2  6008501      +  C  G 135  0.2057095 0.06637371  9.605423
rs2521089          3 10487652      -  T  C 135  0.1775016 0.05758287  9.502064
           P1df Pc1df      effAB      effBB  chi2.2df P2df
rs1719133 0.460 0.525 -0.2080882 -0.7375000 14.729116   NA
rs2975760 0.470 0.555  0.2755102  0.4090909 13.547345   NA
rs7418878 0.835 0.900  0.2807405  0.3268398 12.181064   NA
rs5308595 0.855 0.915  0.2564832  0.4623656 10.766439   NA
rs4804634 0.900 0.945 -0.1193830 -0.3845238 11.200767   NA
rs3224311 0.920 0.965  0.2778634  0.3151515 11.658283   NA
rs26325   0.930 0.965 -0.4875367         NA 10.298495   NA
rs8835506 0.955 0.980  0.2796221  0.3076923 11.513206   NA
rs3925525 0.980 0.990  0.2660834  0.3074627 10.782867   NA
rs2521089 0.980 0.990  0.2254633  0.3396072  9.933387   NA
```

Experimental-wise significance is computed by an empirical procedure, thus we consider P-values ≤ 0.05 to be GW-significant. However, 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 a certain SNP, the corrected P-values become equal to 1 – at this point the order in the list is arbitrary because sorting could not be done.

Summary:

- The `descriptives` family of functions was developed to facilitate the 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-package` is `check.marker()`. We will now use that to perform our data QC; the output is rather self-explaining. Because of possible genetic heterogeneity of the study data it is a good idea to

skip Hardy-Weinberg checks in the first round of QC. 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 )
0 people have intermediate X-chromosome inbreeding (0.5 > F > 0.5)
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 0.7704304 (s.e. 0.02151671), as based on 2000 autosomal markers
2 (1.492537%) people excluded because of too high IBS (>=0.95)
In total, 3653 (91.4851%) markers passed all criteria
In total, 128 (95.52239%) people passed all criteria

RUN 2
3653 markers and 128 people in total
80 (2.189981%) markers excluded as having low (<1.953125%) 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 0.2748341 (s.e. 0.01695461)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7710095 (s.e. 0.01769682), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 3573 (97.81002%) markers passed all criteria
In total, 128 (100%) people passed all criteria

RUN 3
```

```
3573 markers and 128 people in total
0 (0%) markers excluded as having low (<1.953125%) 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 0.2748341 (s.e. 0.01695461)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7699498 (s.e. 0.01725735), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 3573 (100%) markers passed all criteria
In total, 128 (100%) people passed all criteria
```

Note that normally you will *NEVER* run this simple form of the QC function – you should always provide a number of thresholds specific to the platform you used for genotyping. See help to `check.marker()` for detailed list of arguments. The default values used by the function are rather relaxed compared to the thresholds routinely used nowadays with most of the platforms.

The computation of all pairwise proportion of alleles identical-by-state (IBS) by `ibs()` function, which is also called by `check.marker()` 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 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 the pseudoautosomal region, which should have been arranged as a separate "autosome". Nine people are found to have intermediate inbreeding at the X-chromosome and are also excluded from analysis.

Then, the procedure finds the markers with low call rate (≤ 0.95 by default) across people, markers with low MAF (by default, low MAF is defined as less than a few copies of the rare allele, see help for details); people with low call rate (default value: ≤ 0.95) across SNPs, people with extreme heterozygosity (at FDR 0.01) and those who have GW IBS ≥ 0.95. These default parameters may be changed if you wish (consult the 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 having low quality) samples. Therefore the QC is repeated iteratively until no further errors are found.

You can generate a short summary of the 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   384     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 otherSexErr
IDnoCall           1       0       0        0      0     0           0
HetFail           NA       3       0        0      0     0           0
IBSFail           NA      NA       2        0      0     0           0
isfemale          NA      NA      NA        2      0     0           0
ismale            NA      NA      NA       NA      0     0           0
isXXY             NA      NA      NA       NA     NA     0           0
otherSexErr       NA      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"
 [6] "idnocall"    "ibsfail"     "isfemale"    "ismale"      "otherSexErr"
[11] "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 (and should!) 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 128 people left in the 'clean' data set). Therefore we need to detach the data by

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

and attach new data by

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

At this stage, let us check if the first round of QC improves the fit of genetic data to HWE, which may have been violated due to by genotyping errors which we hopefully (at least partly!) eliminated:

```
> 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     44.000   66.000 117.000 239.000  3573
Prop    0.012    0.018   0.033   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  3573
Prop    0.013     0.02   0.036   0.064     1

> descriptives.marker(data1[dm2==0, ])[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          0        0   7.000  91.000  3573
Prop        0        0   0.002   0.025     1
```

You can see that the fit to HWE improved, but cases are still have excess number of markers out of HWE. This may be due to genetic sub-structure.

5.3 Finding genetic sub-structure

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

As the first step, we will compute a matrix of genomic kinship between all pairs of individuals, using only autosomal[2] markers by

```
> data1.gkin <- ibs(data1[, autosomal(data1)], weight="freq")
```

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

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

[2] the list of autosomal markers contained in `data` is returned by the `autosomal(data)` function

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

```
             id199         id300         id403         id415        id666
id199  0.494427766 3255.00000000 3253.00000000 3241.00000000 3257.0000000
id300 -0.011754266    0.49360296 3261.00000000 3250.00000000 3264.0000000
id403 -0.012253378   -0.01262949    0.50541775 3247.00000000 3262.0000000
id415 -0.001812109    0.01388179   -0.02515438    0.53008236 3251.0000000
id666 -0.018745051   -0.02127344    0.02083723   -0.02014175    0.5306584
```

The numbers below the diagonal show the genomic estimate of kinship (aka 'genomic kinship' or 'genome-wide 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 shown 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 those 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))
> if (length(cl1) > length(cl2)) {x<-cl2; cl2<-cl1; cl1<-x}
> 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"
```

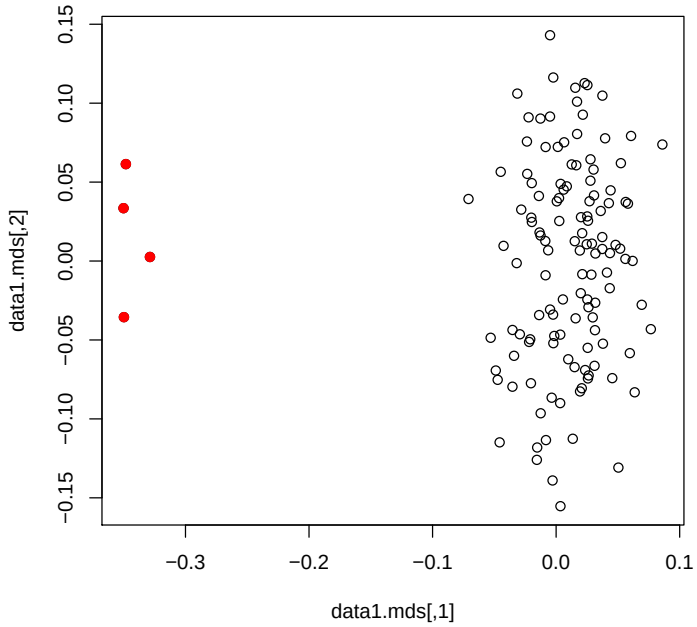

Figure 5.3: Mapping samples to the space of the first two Principle Components resulting from analysis of genomic kinship. Red dots identify genetic outliers.

```
 [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" "id2286"
[121] "id4743" "id4185" "id8330" "id6934"
```

Four outliers are shown 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=(phdata(data2)$dm2==0), fdr=0.2)

Excluding people/markers with extremely low call rate...
3573 markers and 124 people in total
0 people excluded because of call rate < 0.1
0 markers excluded because of call rate < 0.1
Passed: 3573 markers and 124 people

Running sex chromosome checks...
0 heterozygous X-linked male genotypes found
0 X-linked markers are likely to be autosomal (odds > 1000 )
0 male are likely to be female (odds > 1000 )
0 female are likely to be male (odds > 1000 )
0 people have intermediate X-chromosome inbreeding (0.5 > F > 0.5)
If these people/markers are removed, 0 heterozygous male genotypes are left
Passed: 3573 markers and 124 people

no X/Y/mtDNA-errors to fix


RUN 1
3573 markers and 124 people in total
40 (1.119507%) markers excluded as having low (<2.016129%) 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 0.2780246 (s.e. 0.01642372)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7714255 (s.e. 0.0124453), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 3533 (98.88049%) markers passed all criteria
In total, 124 (100%) people passed all criteria

RUN 2
3533 markers and 124 people in total
0 (0%) markers excluded as having low (<2.016129%) 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 0.2780246 (s.e. 0.01642372)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7684757 (s.e. 0.01265441), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 3533 (100%) markers passed all criteria
In total, 124 (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 otherSexErr
IDnoCall           0       0       0        0      0     0           0
HetFail           NA       0       0        0      0     0           0
IBSFail           NA      NA       0        0      0     0           0
isfemale          NA      NA      NA        0      0     0           0
ismale            NA      NA      NA       NA      0     0           0
isXXY             NA      NA      NA       NA     NA     0           0
otherSexErr       NA      NA      NA       NA     NA    NA           0
```

If the procedure did not run, check the previous Note.

Indeed, in the updated data set several 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, so we do not need to type `phdata(data2)$...` to access `dm2` status or other variables:

```
> detach(phdata(data1))
> attach(phdata(data2))
> ####
> #ge03d2ex.clean <- data2
> #save(ge03d2ex.clean, file="ge03d2ex.clean.RData")
> ####
```

Before proceeding to GWA, let us check if complete QC improved the fit of genetic data to HWE:

```
> 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   2e+01 101.000  3533
Prop        0    0.001   6e-03   0.029     1

> descriptives.marker(data2[phdata(data2)$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          0        1  17.000  79.000  3533
Prop        0        0   0.005   0.022     1
```

```
> descriptives.marker(data2[phdata(data2)$dm2==0, ])[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          0        0   7.000  91.000  3533
Prop        0        0   0.002   0.026     1
```

You can see that now there is no excessive number of SNPs out of HWE in the sample (total, or cases, or controls)

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               47      NA     NA           77      NA     NA    NA    NA
sex              47   0.426  0.500           77   0.597  0.494 0.065 0.064
age              47  45.752 13.313           77  50.593 12.465 0.047 0.062
dm2              47      NA     NA           77      NA     NA    NA    NA
height           46 167.911  8.689           77 170.423 10.646 0.157 0.213
weight           46  77.015 17.528           77  94.160 26.963 0.000 0.000
diet             47   0.064  0.247           77   0.065  0.248 0.981 0.981
bmi              46  27.424  6.598           77  32.235  8.335 0.001 0.001
       Pexact
id         NA
sex     0.067
age        NA
dm2        NA
height     NA
weight     NA
diet    1.000
bmi        NA
```

You can see that the relation to weight is maintained in this smaller, but hopefully cleaner, data set; moreover, the relation to age becomes borderline 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   2e+01 101.000  3533
Prop        0    0.001   6e-03   0.029     1
```

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

Run the score test on the cleaned data by

```
> data2.qt <- qtscore(dm2, data2, trait="binomial")
```

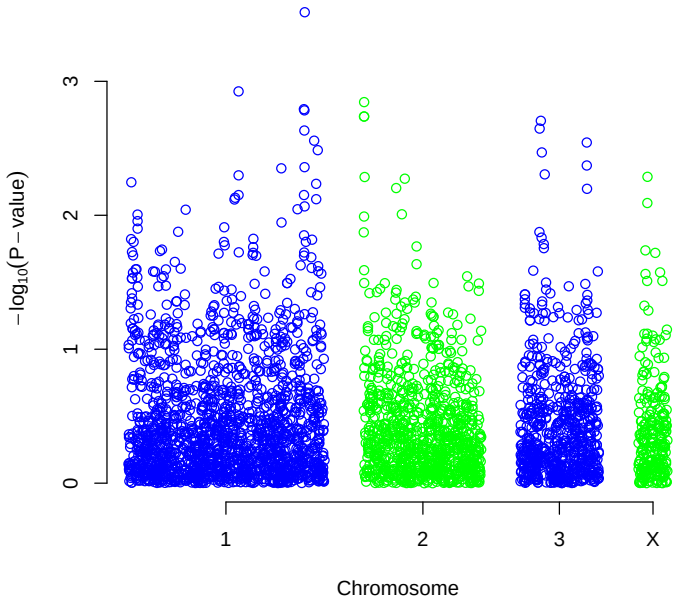

Figure 5.4: $-\log_{10}$(Corrected P-value) from the genome scan after the QC procedure.

and check lambda

```
> lambda(data2.qt)

$estimate
[1] 1.036958

$se
[1] 0.0007178
```

there is still some inflation, which is explained by the fact that we investigate only a 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")

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2   N      effB    se_effB  chi2.1df
rs1719133          1  4495479      +  T  A 124 0.3167801 0.08614528 13.522368
```

```
rs4804634          1  2807417      +  C  G 121 0.4119844 0.12480696 10.896423
rs8835506          2  6010852      +  A  T 121 3.5378209 1.08954331 10.543448
rs4534929          1  4474374      +  C  G 123 0.4547151 0.14160410 10.311626
rs1013473          1  4487262      +  A  T 124 2.7839368 0.86860745 10.272393
rs3925525          2  6008501      +  C  G 124 3.2807631 1.03380675 10.070964
rs3224311          2  6009769      +  G  C 124 3.2807631 1.03380675 10.070964
rs2975760          3 10518480      +  A  T 123 3.1802120 1.00916993  9.930784
rs2521089          3 10487652      -  T  C 123 2.7298775 0.87761175  9.675679
rs1048031          1  4485591      +  G  T 122 0.4510793 0.14548378  9.613391
                  P1df     effAB     effBB  chi2.2df         P2df        Pc1df
rs1719133 0.0002357368 0.3740771 0.0000000 14.677906 0.0006497303 0.0003048399
rs4804634 0.0009635013 0.6315789 0.1739130 12.375590 0.0020543516 0.0011885463
rs8835506 0.0011660066 4.0185185 4.0185185 12.605556 0.0018312105 0.0014292471
rs4534929 0.0013219476 0.4830918 0.1739130 10.510272 0.0052206352 0.0016136479
rs1013473 0.0013503553 3.0495868 5.8441558 10.926296 0.0042401869 0.0016471605
rs3925525 0.0015062424 3.6923077 4.0000000 11.765985 0.0027864347 0.0018306610
rs3224311 0.0015062424 3.6923077 4.0000000 11.765985 0.0027864347 0.0018306610
rs2975760 0.0016253728 3.0000000 8.0000000 10.172522 0.0061810866 0.0019704699
rs2521089 0.0018672326 3.0147059 5.0000000 10.543296 0.0051351403 0.0022533033
rs1048031 0.0019316360 0.4844720 0.1714286  9.965696 0.0068545128 0.0023284084
```

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 assessing empirical GW significance:

```
> data2.qte <- qtscore(dm2,
+                      data2, times=200, quiet=TRUE, trait="binomial")

  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

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

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2   N      effB    se_effB  chi2.1df
rs1719133          1  4495479      +  T  A 124 0.3167801 0.08614528 13.522368
rs4804634          1  2807417      +  C  G 121 0.4119844 0.12480696 10.896423
rs8835506          2  6010852      +  A  T 121 3.5378209 1.08954331 10.543448
rs4534929          1  4474374      +  C  G 123 0.4547151 0.14160410 10.311626
rs1013473          1  4487262      +  A  T 124 2.7839368 0.86860745 10.272393
rs3925525          2  6008501      +  C  G 124 3.2807631 1.03380675 10.070964
rs3224311          2  6009769      +  G  C 124 3.2807631 1.03380675 10.070964
rs2975760          3 10518480      +  A  T 123 3.1802120 1.00916993  9.930784
rs1048031          1  4485591      +  G  T 122 0.4510793 0.14548378  9.613391
rs2521089          3 10487652      -  T  C 123 2.7298775 0.87761175  9.675679
           P1df Pc1df     effAB     effBB  chi2.2df  P2df
rs1719133 0.345 0.425 0.3740771 0.0000000 14.677906 0.465
rs4804634 0.800 0.865 0.6315789 0.1739130 12.375590 0.945
rs8835506 0.860 0.940 4.0185185 4.0185185 12.605556 0.910
```

```
rs4534929 0.910 0.955 0.4830918 0.1739130 10.510272 1.000
rs1013473 0.910 0.955 3.0495868 5.8441558 10.926296 0.995
rs3925525 0.950 0.960 3.6923077 4.0000000 11.765985 0.975
rs3224311 0.950 0.960 3.6923077 4.0000000 11.765985 0.975
rs2975760 0.955 0.960 3.0000000 8.0000000 10.172522 1.000
rs1048031 0.960 0.980 0.4844720 0.1714286  9.965696 1.000
rs2521089 0.960 0.980 3.0147059 5.0000000 10.543296 1.000
```

Again, none of the SNPs hits GW 5% significance. Still, you can see that after QC the 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=TRUE, trait="binomial")

  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

> descriptives.scan(data2.qtae)

Summary for top 10 results, sorted by P1df
          Chromosome Position Strand A1 A2   N      effB   se_effB  chi2.1df
rs1719133          1  4495479      +  T  A 124 0.5057834 0.1386464 13.307976
rs2398949          1  4828375      -  A  C 122 0.4163599 0.1253217 11.037861
rs4804634          1  2807417      +  C  G 121 0.5811444 0.1769232 10.789433
rs7522488          3 11689797      -  G  A 123 1.8718644 0.5707656 10.755565
rs3925525          2  6008501      +  C  G 124 2.0202686 0.6406356  9.944799
rs3224311          2  6009769      +  G  C 124 2.0202686 0.6406356  9.944799
rs8835506          2  6010852      +  A  T 121 2.0575431 0.6501899 10.014228
rs1037237          3 11690145      +  C  G 124 1.8144282 0.5735660 10.007206
rs1013473          1  4487262      +  A  T 124 1.8148181 0.5864622  9.576043
rs1048031          1  4485591      +  G  T 122 0.6128584 0.1993057  9.455418
           P1df Pc1df     effAB     effBB  chi2.2df  P2df
rs1719133 0.470 0.595 0.4853657 0.2242383 13.367455 0.890
rs2398949 0.845 0.915 0.3110042 2.8197228 16.111010 0.355
rs4804634 0.870 0.925 0.7420448 0.3389525 12.112794 0.990
rs7522488 0.875 0.930 1.3995557 3.1160361 11.256933 0.995
rs3925525 0.940 1.000 2.3028971 2.2451502 11.988964 0.990
rs3224311 0.940 1.000 2.3028971 2.2451502 11.988964 0.990
rs8835506 0.940 0.995 2.3758119 2.2365358 12.302956 0.990
rs1037237 0.940 0.995 1.3400938 2.9836475 10.674189 1.000
rs1013473 0.980 1.000 1.8390955 2.8613308  9.893424 1.000
rs1048031 0.995 1.000 0.6453091 0.3417444  9.840158 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 ≥ 30) cases to all controls.

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

  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

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

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2  N      effB   se_effB   chi2.1df
rs7522488          3 11689797      -  G  A 88 1.7561171 0.5560485 9.97428216
rs1037237          3 11690145      +  C  G 88 1.7561171 0.5560485 9.97428216
rs9630764          1  3897972      +  T  A 88 1.7595598 0.5615384 9.81858908
rs1891586          1  2297398      -  C  T 88 0.5340528 0.1733886 9.48696549
rs3215698          X 13559835      -  T  A 88 0.4806715 0.1571512 9.35537756
rs1646456          1      653      +  C  G 87 0.8853536 1.3981894 0.40096049
rs4435802          1     5291      +  C  A 86 1.7961795 1.1926370 2.26820816
rs946364           1     8533      -  T  C 86 0.7682453 0.5994013 1.64272311
rs299251           1    10737      +  A  G 88 0.6819981 0.8106956 0.70770257
rs2456488          1    11779      +  G  C 87 0.9613536 4.4351609 0.04698374
           P1df Pc1df     effAB     effBB   chi2.2df P2df
rs7522488 0.910 0.925 1.3215457 3.1245402 11.2990479 0.96
rs1037237 0.910 0.925 1.3215457 3.1245402 11.2990479 0.96
rs9630764 0.925 0.940 1.9346110 3.3161548  9.8316163 1.00
rs1891586 0.970 0.975 0.5836292 0.3065275  9.5182135 1.00
rs3215698 0.975 0.985 0.3406693 0.2701336  9.9417133 1.00
rs1646456 1.000 1.000 1.1455730 0.6013127  2.2015462 1.00
rs4435802 1.000 1.000 1.7919867        NA  2.2682082 1.00
rs946364  1.000 1.000 0.8791195 0.4696496  2.1881175 1.00
rs299251  1.000 1.000 0.6835419        NA  0.7077026 1.00
rs2456488 1.000 1.000 1.0713560 0.8544801  0.3319582 1.00
```

Again, nothing interesting at the GW significance level. If we would have had found something, naturally, we would not have 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).

Let us save the 'data1', 'data1.gkin', 'data2.qt' and 'cl1' objects now

```
> save(data1, cl1, data1.gkin, data2.qt, file="data1.RData")
```

These data will be used later in section 7, "GWA in presence of genetic stratification: practice", in which we will perform GWA using different methods to account for stratification, and will compare the results with these obtaining by removing outliers ('data2.qt').

Let us also save the cleaned 'data2' object, which will be later used in section 10 ("Meta-analysis of GWA scans"):

```
> save(data2, file="data2.RData")
```

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 Genome-wide association analysis exercise

During the exercise, you will work with a larger data set (approximately 1,000 people and 7,000+ SNPs). You are to do the 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 known 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-?.?.?". Load the `GenABEL-package` library by

```
> library(GenABEL)
```

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

```
> data(ge03d2)
```

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

Ex. 1 — How many cases and controls are present in the original data set?

Ex. 2 — How many markers are present in the original data set?

Ex. 3 — Is there evidence for inflation of the HWE test statistics?

Ex. 4 — Analyse empirical GW significance. How many SNPs pass the genome-wide significance threshold, after correction for the inflation factor? Write down the names of these SNPs for further comparison.

Ex. 5 — Perform first steps of the genetic data QC.

Ex. 6 — How many males are 'genetically' females?

Ex. 7 — How many females are 'genetically' males?

Ex. 8 — How many people are quessed to have 'XXY' genotype?

Ex. 9 — 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(s)` these

Ex. 10 — How many "twin" DNAs did you discover?

Ex. 11 — Perform second step of QC.

Ex. 12 — How many genetic outliers did you discover?

Ex. 13 — How many cases and controls are presented in the data after QC?

Ex. 14 — How many markers are presented in the data after QC?

Ex. 15 — Is there evidence for inflation of the HWE test staistics?

Ex. 16 — Perform GWA analysis of the cleaned data, using asimptotic test and plot the results. What is the estimate of λ for the 1 d.f. test?

Ex. 17 — Analyse empirical GW significance. How many SNPs pass genome-wide significance threshold, after correction for the inflation factor?

Ex. 18 — Do these SNPs overlap much with the ones ranked at the top before the QC? If not, what could be the reason?

Ex. 19 — 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`.

Ex. 20 — Given the two-stage design, and applying the puristic criteria specified in the lecture, for how many SNPs you can claim a significant finding?

Ex. 21 — Using the same criteria, for how many SNPs you can claim a replicated finding?

Ex. 22 — If time permits, characterise the mode of inheritance of the significant SNPs. You can convert data from `GenABEL-package` format to the format used by `dgc.genetics` and `genetics` libraries by using `as.genotype()` function. Consult the help for details. Please do not attempt to convert more than a few dozen 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.

Ex. 23 — If time permits, do analysis with adjustment for covariates and stratified analysis.

Ex. 24 — 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?

5.6 Answers to exercises

Answer (Ex. 1) — :

```
> table(phdata(ge03d2)$dm2)

  0   1
487 463
```

Answer (Ex. 2) — :

```
> nsnps(ge03d2)

[1] 7589
```

Answer (Ex. 3) — Yes:

```
> descriptives.marker(ge03d2)[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    331.000  367.000 479.000 807.000  7589
Prop    0.044    0.048   0.063   0.106     1
```

Answer (Ex. 4) — :

```
> res0 <- qtscore(dm2, data=ge03d2, times=200,
+                 quiet=TRUE, trait="binomial")
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
> ### something funny is going on here
> ### disabeling next line
> #lambda(res0)
> ds <- descriptives.scan(res0)

Summary for top 10 results, sorted by P1df

> ds
          Chromosome Position Strand A1 A2   N effB se_effB chi2.1df P1df Pc1df
rs1646456          1      653      +  C  G 938  NaN     NaN      NaN   NA    NA
rs7950586          1      849      -  T  A 938  NaN     NaN      NaN   NA    NA
rs4785242          1     1766      -  T  C 936  NaN     NaN      NaN   NA    NA
rs4435802          1     5291      +  C  A 943  NaN     NaN      NaN   NA    NA
rs2847446          1     5555      +  T  A 937  NaN     NaN      NaN   NA    NA
rs9308393          1     6739      +  T  C 937  NaN     NaN      NaN   NA    NA
rs946364           1     8533      -  T  C 938  NaN     NaN      NaN   NA    NA
rs299251           1    10737      +  A  G 942  NaN     NaN      NaN   NA    NA
rs2456488          1    11779      +  G  C 934  NaN     NaN      NaN   NA    NA
rs1292700          1    12710      -  A  C 941  NaN     NaN      NaN   NA    NA
              effAB         effBB chi2.2df P2df
rs1646456 1.1117443           NaN        0   NA
rs7950586       NaN           NaN        0   NA
```

```
rs4785242 0.8842832           NaN        0   NA
rs4435802       NaN 6.970168e-309        0   NA
rs2847446       NaN           NaN        0   NA
rs9308393       NaN            NA        0   NA
rs946364        NaN  9.283721e-01        0   NA
rs299251        NaN           NaN        0   NA
rs2456488       NaN           NaN        0   NA
rs1292700       NaN  8.457300e-01        0   NA
```

Thus, there are no genome-wide empirically significant results. The 'top' 10 SNPs are

```
> snps0 <- rownames(ds)
> snps0
 [1] "rs1646456" "rs7950586" "rs4785242" "rs4435802" "rs2847446" "rs9308393"
 [7] "rs946364"  "rs299251"  "rs2456488" "rs1292700"
```

(note that if the empirical $P = 1$, the rank is assigned quite arbitrarily)

Answer (Ex. 5) — First step of QC

```
> qc1 <- check.marker(ge03d2, call=0.95, perid.call=0.95,
+                     p.level=0, ibs.exclude="both")
Excluding people/markers with extremely low call rate...
7589 markers and 950 people in total
0 people excluded because of call rate < 0.1
7 markers excluded because of call rate < 0.1
Passed: 7582 markers and 950 people

Running sex chromosome checks...
1934 heterozygous X-linked male genotypes found
2 X-linked markers are likely to be autosomal (odds > 1000 )
10 male are likely to be female (odds > 1000 )
6 female are likely to be male (odds > 1000 )
0 people have intermediate X-chromosome inbreeding (0.5 > F > 0.5)
If these people/markers are removed, 8 heterozygous male genotypes are left
... these will be considered missing in analysis.
... Use Xfix() to fix these problems.
Passed: 7580 markers and 934 people

... 8 X/Y/mtDNA ( 8 0 0 ) impossible heterozygotes and female Ys set as missing


RUN 1
7580 markers and 934 people in total
73 (0.9630607%) markers excluded as having low (<0.267666%) minor allele frequency
75 (0.9894459%) markers excluded because of low (<95%) call rate
0 (0%) markers excluded because they are out of HWE (P <0)
4 (0.4282655%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2558271 (s.e. 0.02102863)
4 (0.4282655%) people excluded because too high autosomal heterozygosity (FDR <1%)
```

```
Excluded people had HET >= 0.4702949
Mean IBS is 0.7852411 (s.e. 0.01742783), as based on 2000 autosomal markers
8 (0.856531%) people excluded because of too high IBS (>=0.95)
In total, 7432 (98.04749%) markers passed all criteria
In total, 918 (98.28694%) people passed all criteria

RUN 2
7432 markers and 918 people in total
42 (0.5651238%) markers excluded as having low (<0.2723312%) 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 0.2562716 (s.e. 0.0151377)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7879519 (s.e. 0.01554664), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 7390 (99.43488%) markers passed all criteria
In total, 918 (100%) people passed all criteria

RUN 3
7390 markers and 918 people in total
0 (0%) markers excluded as having low (<0.2723312%) 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 0.2562716 (s.e. 0.0151377)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7864124 (s.e. 0.01618755), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 7390 (100%) markers passed all criteria
In total, 918 (100%) people passed all criteria
```

```
> summary(qc1)

$`Per-SNP fails statistics`
          NoCall NoMAF NoHWE Redundant Xsnpfail
NoCall        82     0     0         0        0
NoMAF         NA   115     0         0        0
NoHWE         NA    NA     0         0        0
Redundant     NA    NA    NA         0        0
Xsnpfail      NA    NA    NA        NA        2

$`Per-person fails statistics`
            IDnoCall HetFail IBSFail isfemale ismale isXXY otherSexErr
IDnoCall           4       0       0        0      0     0           0
HetFail           NA       4       0        0      0     0           0
IBSFail           NA      NA       8        0      0     0           0
isfemale          NA      NA      NA       10      0     0           0
ismale            NA      NA      NA       NA      6     0           0
isXXY             NA      NA      NA       NA     NA     0           0
otherSexErr       NA      NA      NA       NA     NA    NA           0
```

```
> data1 <- ge03d2[qc1$idok, qc1$snpok]
> data1 <- Xfix(data1)
... 7 X/Y/mtDNA ( 7 0 0 ) impossible heterozygotes and female Ys set as missing
> qc2 <- check.marker(data1, call=0.95, perid.call=0.95, p.level=0)
Excluding people/markers with extremely low call rate...
7390 markers and 918 people in total
0 people excluded because of call rate < 0.1
0 markers excluded because of call rate < 0.1
Passed: 7390 markers and 918 people

Running sex chromosome checks...
0 heterozygous X-linked male genotypes found
0 X-linked markers are likely to be autosomal (odds > 1000 )
0 male are likely to be female (odds > 1000 )
0 female are likely to be male (odds > 1000 )
0 people have intermediate X-chromosome inbreeding (0.5 > F > 0.5)
If these people/markers are removed, 0 heterozygous male genotypes are left
Passed: 7390 markers and 918 people

no X/Y/mtDNA-errors to fix


RUN 1
7390 markers and 918 people in total
0 (0%) markers excluded as having low (<0.2723312%) 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 0.2562716 (s.e. 0.0151377)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7908779 (s.e. 0.01581427), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 7390 (100%) markers passed all criteria
In total, 918 (100%) people passed all criteria
> summary(qc2)
$`Per-SNP fails statistics`
          NoCall NoMAF NoHWE Redundant Xsnpfail
NoCall         0     0     0         0        0
NoMAF         NA     0     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 otherSexErr
IDnoCall           0       0       0        0      0     0           0
HetFail           NA       0       0        0      0     0           0
IBSFail           NA      NA       0        0      0     0           0
```

```
isfemale          NA      NA      NA       0      0     0          0
ismale            NA      NA      NA      NA      0     0          0
isXXY             NA      NA      NA      NA     NA     0          0
otherSexErr       NA      NA      NA      NA     NA    NA          0
```

Answer (Ex. 6) — The list of genetic females who are coded as males is

```
> qc1$isfemale
 [1] "id3374" "id6263" "id6835" "id8410" "id8509" "id8519" "id8542" "id2701"
 [9] "id6494" "id3100"
```

Answer (Ex. 7) — The list of genetic males who are coded as females is

```
> qc1$ismale
[1] "id193"  "id8475" "id2461" "id5669" "id7245" "id8301"
```

Answer (Ex. 8) — The number of 'XXY' people is 0

Answer (Ex. 9) — Eight 'sporadic' X-errors are left after removing people with likely sex code errors (seven in the data set after first step of QC)

Answer (Ex. 10) — The list of IDs failing IBS checks ('twin' DNAs) is

```
> qc1$ibsfail
[1] "id3368" "id9668" "id5437" "id956"  "id386"  "id660"  "id2115" "id8370"
```

Answer (Ex. 11) — The second step of QC:

```
> data1.gkin <- ibs(data1[, autosomal(data1)], weight="freq")
> data1.dist <- as.dist(0.5 - data1.gkin)
> data1.mds <- cmdscale(data1.dist)
> km <- kmeans(data1.mds, centers=2, nstart=1000)
> cl1 <- names( which(km$cluster==1) )
> cl2 <- names( which(km$cluster==2) )
> if (length(cl1) > length(cl2)) {x<-cl2; cl2<-cl1; cl1<-x}
> cl1
 [1] "id2097" "id2126" "id2878" "id3021" "id3176" "id4554" "id4756" "id7436"
 [9] "id7533" "id9396" "id9546" "id4021" "id2171" "id6954" "id2136" "id5056"
[17] "id1751" "id6626" "id2970" "id1300" "id8639" "id1729" "id9398" "id9904"
[25] "id858"
> data2 <- data1[cl2,]
> qc2 <- check.marker(data2, hweids=(phdata(data2)$dm2==0), fdr=0.2)
Excluding people/markers with extremely low call rate...
7390 markers and 893 people in total
0 people excluded because of call rate < 0.1
```

```
0 markers excluded because of call rate < 0.1
Passed: 7390 markers and 893 people

Running sex chromosome checks...
0 heterozygous X-linked male genotypes found
0 X-linked markers are likely to be autosomal (odds > 1000 )
0 male are likely to be female (odds > 1000 )
0 female are likely to be male (odds > 1000 )
0 people have intermediate X-chromosome inbreeding (0.5 > F > 0.5)
If these people/markers are removed, 0 heterozygous male genotypes are left
Passed: 7390 markers and 893 people

no X/Y/mtDNA-errors to fix


RUN 1
7390 markers and 893 people in total
5 (0.067659%) markers excluded as having low (<0.2799552%) 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 0.2565083 (s.e. 0.01505982)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.787009 (s.e. 0.01174357), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 7385 (99.93234%) markers passed all criteria
In total, 893 (100%) people passed all criteria

RUN 2
7385 markers and 893 people in total
0 (0%) markers excluded as having low (<0.2799552%) 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 0.2565083 (s.e. 0.01505982)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7902995 (s.e. 0.01171197), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 7385 (100%) markers passed all criteria
In total, 893 (100%) people passed all criteria
```

> summary(qc2)

```
$`Per-SNP fails statistics`
          NoCall NoMAF NoHWE Redundant Xsnpfail
NoCall         0     0     0         0        0
NoMAF         NA     5     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 otherSexErr
IDnoCall           0       0       0        0      0     0           0
HetFail           NA       0       0        0      0     0           0
IBSFail           NA      NA       0        0      0     0           0
isfemale          NA      NA      NA        0      0     0           0
ismale            NA      NA      NA       NA      0     0           0
isXXY             NA      NA      NA       NA     NA     0           0
otherSexErr       NA      NA      NA       NA     NA    NA           0
```

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

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

```
> ####
> #ge03d2.clean <- data2
> #save(ge03d2.clean, file="ge03d2.clean.RData")
> ####
```

Answer (Ex. 12) — The list of genetic outliers is

```
> cl1
```

```
 [1] "id2097" "id2126" "id2878" "id3021" "id3176" "id4554" "id4756" "id7436"
 [9] "id7533" "id9396" "id9546" "id4021" "id2171" "id6954" "id2136" "id5056"
[17] "id1751" "id6626" "id2970" "id1300" "id8639" "id1729" "id9398" "id9904"
[25] "id858"
```

Answer (Ex. 13) — :

```
> table(phdata(data2)$dm2)
```

```
  0   1
472 421
```

Answer (Ex. 14) — :

```
> nsnps(data2)
```

```
[1] 7385
```

Answer (Ex. 15) — No:

```
> 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    4.000  44.000 258.000  7385
Prop        0    0.001   0.006   0.035     1
```

FROM THIS POINT ON THE RUNNING OF THE TUTORIAL FAILS (2013.02.20, USING GENABEL DEV VERSION 1.7-4). I COMMENTED OUT THE R CODE BELOW.

Answer (Ex. 16) — :

```
> #qts <- qtscore(dm2, data2, trait="binomial")
> #lambda(qts)
```

Answer (Ex. 17) — :

```
> #res1 <- qtscore(dm2, data=data2, times=200, quiet=TRUE, trait="binomial")
> #ds1 <- descriptives.scan(res1)
> #ds1
```

There are SNPs which are empirically genome-wide significant in the data. To get the list of 'top' 10 SNPs:

```
> #snps1 <- rownames(ds1)
> #snps1
```

Answer (Ex. 18) — There is little overlap between SNPs before and after QC:

```
> #snps0
> #snps1
```

Answer (Ex. 19) — :

```
> #data(ge03d2c)
> #snps1
> #confdat <- ge03d2c[, snps1]
> #rep <- qtscore(dm2, confdat, times=10000, quiet=TRUE)
> #descriptives.scan(rep)
```

Answer (Ex. 20) — Two-stage P-value is

```
> #    snps1
> #    finres <- matrix(NA, 10, 3)
> #    colnames(finres) <- c("Stage 1", "Replication", "Combined")
> #    rownames(finres) <- snps1
> #    for (i in 1:10) {
> #        finres[i, 1] <- res1[which(snpnames(data2)==snps1[i]), "Pc1df"]
> #        finres[i, 2] <- rep[which(snpnames(confdat)==snps1[i]), "P1df"]
> #        finres[i, 3] <- finres[i, 1]*finres[i, 2]
> #    }
> #    finres
> #    for (i in 1:10) {
> #        if (finres[i, 3] <= 0.05) {
> #            print(c("---------", rownames(finres)[i], "-------"))
> #
> #            print(c(rownames(finres)[i], "stage 1:"))
> #            ph <- phdata(data2)$dm2
> #            gt <- as.numeric(data2[, rownames(finres)[i]])
```

```
> #            print(summary( glm(ph~gt, family=binomial) )$coef)
> #
> #            print(c(rownames(finres)[i], "stage 2:"))
> #            ph <- phdata(confdat)$dm2
> #            gt <- as.numeric(confdat[, rownames(finres)[i]])
> #            print(summary(glm(ph~gt, family=binomial))$coef)
> #
> #            print(c(rownames(finres)[i], "Joint:"))
> #            ph <- c(phdata(data2)$dm2, phdata(confdat)$dm2)
> #            gt <- c(as.numeric(data2[, rownames(finres)[i]]),
> #                    as.numeric(confdat[, rownames(finres)[i]]))
> #            print(summary( glm(ph~gt, family=binomial) )$coef)
> #        }
> #    }
> #replicatedsnps <- rownames(finres)[finres[, "Stage 1"] <= 0.05 &
> #                                   finres[, "Replication"] <= 0.05 &
> #                                   finres[, "Combined"] <= 0.05]
> #replicatedsnps
> #sigsnps <- rownames(finres)[finres[, "Combined"] <= 0.05]
> #sigsnps
```

At the first glance, NWSexprlength(replicatedsnps) SNPs may be claimed as replicated because both first stage and replication P-values are ≤ 0.05 and effects are consistent, and additionally NWSexprlength(sigsnps) - length(replicatedsnps) may be claimed as 'significant' because joint P-values are ≤ 0.05 and the effects are consistent. Generally, a more thorough simulation experiment should be performed.

Answer (Ex. 21) — SNP "rs7903146" had empirical P-value ≤ 0.05 at both stages, and very strong joint significance. It can be claimed as replicated.
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 some are located in known T2D genes (just because we used these names to name the "significant" ones).

Chapter 6

GWA analysis in presence of stratification: theory

In genetic association studies, we look for association between a genetic polymorphism and the value of a trait of interest. The best scenario – the one we always hope for – is that the observed association results from causation, that is the polymorphism studied is functionally involved in the control of the trait. However, association has no direction, and making causal inference in epidemiology in general and in genetic epidemiology in particular is usually not possible based on statistical analysis only.

In fact, most associations observed in genetic studies are due to a confounder – an (unobserved) factor which is associated with both the genetic polymorphism and the trait analysed. Presence of such factor leads to induced, "secondary" correlation between the trait and the polymorphism; if we would have controlled for that factor in the association model, the relation between the polymorphism and the trait would have gone.

There are two major types of confounders leading to induced correlation in genetic association studies. One type is "good" confounding of association by the real, unobserved functional variant, which is, as a rule, not present on the SNP array, but is in linkage disequilibrium (LD) with typed SNP. Under this scenario, the functional variant is associated with the trait because of causative relation; at the same time it is associated with a typed polymorphism located nearby because of LD. This confounding induces secondary correlation between the typed polymorphism and the trait, making localistion of the true functional polymorphism (LD mapping) possible.

Other major type of confounding observed in genetic association studies is confounding by population (sub)structure. Let us consider a study in which subjects come from two distantly related populations, say Chinese and European. Due to genetic drift, these two populations will have very different frequencies at many loci throughout the genome. At the same time, these two populations are different phenotypically (prevalence of different disease, mean value of quantitative traita) due to accumulated genetic and cultural differences. Therefore any of these traits will show association with multiple genomic loci. While some of these associations may be genuine genetic associations in a sense that either the polymorphisms themselves, or the polymorphisms close by are causally

involved, most of these associations will be genetically false positives – noise associations generated by strong genetic and phenotypic divergence between the two populations.

The scenario described above is extreme and indeed it is hard to imagine a genetic asociation study in which two very distinct populations are so blintly mixed and analysed not taking this mixture into account. However, a more subtle scenario where several slighly genetically different populations are mixed in the same study is frequently the case and a matter of concern in GWA studies.

In this chapter, we will define what is genetic structure, and how it can be quantified (section **??**); what are the effects of genetic structure on the standard association tests (section **??**) and specific association tests which take possible genetic structure into account (section **??**).

The rest of this chapter is ?temporarily? deleted due to potential copyright issues

Chapter 7

GWA in presence of genetic stratification: practice

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 sections.

7.1 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.

Let us do structured association analysis using the `data1` data derived in previous section.

If you are not running R yet, start R and load `GenABEL-package` library by typing

```
> library(GenABEL)
```

and load the 'data1' workspace generated in section 5 ("Genome-wide association analysis").

```
> load("data1.RData")
> ls()
```

```
[1] "cl1"        "data1"      "data1.gkin" "data2.qt"   "old"
```

First, let us check how much test statistic inflation is there if we ignore stratification.

```
> data1.qt <- qtscore(dm2,data1)
> lambda(data1.qt)
```

```
$estimate
[1] 1.051744

$se
[1] 0.000756613
```

We now will consider several ways to account for stratification, namely, strucutured association analysis, method of Price et al. (EIGENSTRAT), a similar method based on adjusting for the principal components of variation of genomic kinship matrix, and use of a mixed model.

One of the ways to do that is to perform *structured association* analysis. In such analysis, effect and its variance are 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.

To do structured association analysis we need to define a 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(idnames(data1) %in% cl1)
```

Let us check how the 'population' is distributed among the cases and controls:

```
> table(pop,phdata(data1)$dm2)

pop  0  1
  0 47 77
  1  0  4
```

As we have seen before, one of the clusters contains only the cases.

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

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

$estimate
[1] 1.03431

$se
[1] 0.0007059588
```

We can compare the original results, results of analysis excluding outliers, and structured association analysis by

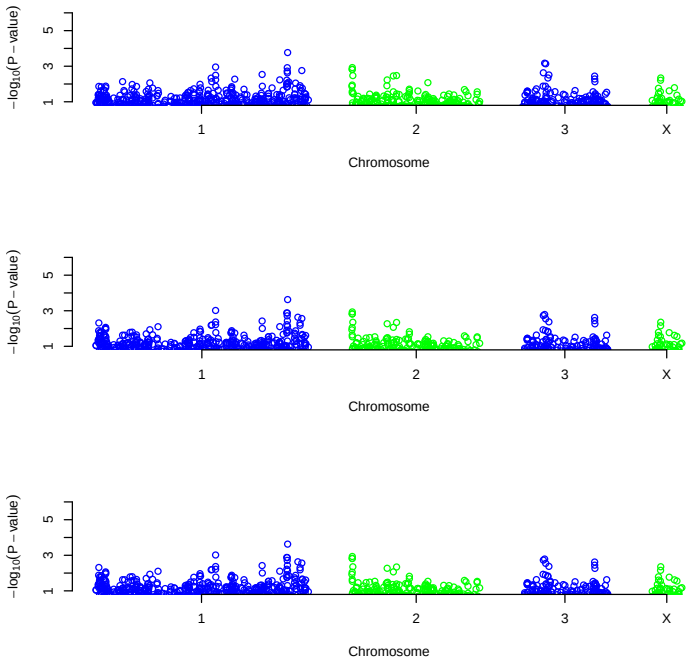

Figure 7.1: Comparison of the original results, results of analysis excluding outliers, and structured association analysis.

```
> par(mfcol=c(3,1))
> plot(data1.qt,ylim=c(1,6))
> plot(data2.qt,ylim=c(1,6))
> plot(data1.sa,ylim=c(1,6))
> par(mfcol=c(1,1))
```

The resulting plot is presented at figure 7.1. In this case, there is 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. (*EIGENSTRAT*), which make use of principal components of the genomic kinship matrix to adjust both phenotypes and genotypes for possible stratification. In `GenABEL-package`, such analysis is done using `egscore` function.

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

$estimate
[1] 1.102747

$se
[1] 0.001043038
```

The analysis plot may be added to the previous one by

```
> par(mfcol=c(3,1))
> plot(data1.eg,ylim=c(1,6))
```

Now let us apply *adjustment for the stratification by use of the principal components of genetic variation*. For that we first need to extract the principal components of genetic variation by constructing the distance matrix

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

and performing the classical multidimensional scaling

```
> pcs <- cmdscale(dst,k=10)
> pcs[1:5,]
```

```
              [,1]         [,2]        [,3]         [,4]        [,5]
id199  0.030289769  0.057952824  0.10342206 -0.020496319 -0.02678162
id300  0.031798583  0.004789923  0.02947431  0.033227508  0.05977593
id403  0.060654557  0.079249013 -0.06111279 -0.081417523  0.03294180
id415 -0.004913697 -0.030714900  0.07807686  0.017468144  0.03078079
id666 -0.012639670  0.016176179 -0.08784632 -0.002322116  0.02957059
             [,6]        [,7]         [,8]         [,9]        [,10]
id199  0.08021104  0.01144582 -0.020228061 -0.001507913 -0.060483058
id300  0.03702791  0.02887831  0.005142389 -0.005340720  0.094991803
id403 -0.01481834 -0.01259170  0.040037267 -0.048689378  0.007304146
id415 -0.02472941  0.07900292  0.075300986  0.028609310  0.021747288
id666 -0.04742543 -0.11594090  0.106539735  0.026380262 -0.022472669
```

Now we can use these PCs for adjustment:

```
> data1.pca <- qtscore(dm2~pcs[,1]+pcs[,2]+pcs[,3],data1)
> lambda(data1.pca)
```

```
$estimate
[1] 1.004946

$se
[1] 0.0007356663
```

```
> plot(data1.pca,ylim=c(1,6))
```

Finally, let us use the full genomic kinship matrix for the adjustemnt for pupulational structure. First, let us estimate the polygenic model with

```
> h2a <- polygenic(dm2,data1,kin=data1.gkin)
```

The resulting 'heritability' estimate is

```
> h2a$esth2
```

```
[1] 0.2547813
```

Now we can perform mixed model approximation analysis using `mmscore` function

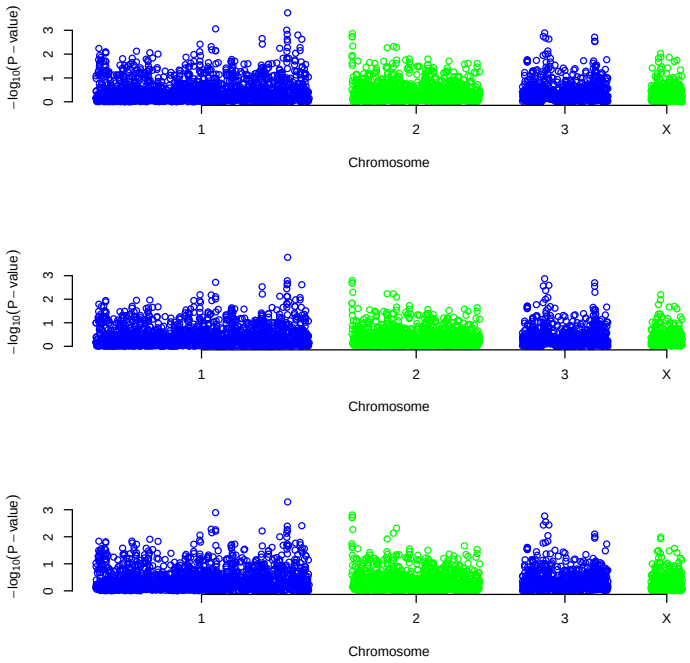

Figure 7.2: Comparison of method of Price et al., principal components adjustment and mixed modelling.

```
> data1.mm <- mmscore(h2a,data1)
> lambda(data1.mm)

$estimate
[1] 1

$se
[1] NA

> plot(data1.mm,ylim=c(1,6))
> par(mfcol=c(1,1))
```

The resulting plot is presented at figure 7.2.

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.

Load and analyse the data set presented in file `stratified.RData`. GWA data presented in this file concern a study containing data from several populations. 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. Answer the questions

DATAPROBLEM: ok, here is a problem; replacing the 'strdat' with data set available within GenABEL. Hence comments to the analysis below are not making sense anymore

Ex. 1 — What covariates are significantly associated with the traits?

Ex. 2 — How many SNPs and IDs are presented in the data set?

Ex. 3 — How many SNPs and IDs pass the quality control (use SNP and ID call rate of 0.98)?

Ex. 4 — Is there evidence for stratification coming from the distribution of GW test for HWE (what is λ?)

Ex. 5 — Is there evidence that the test statistics for trait `quat` is inflated (what is λ?)

Ex. 6 — Is there evidence that the test statistics for trait `bint` is inflated (what is λ?)

Ex. 7 — How many genetically distinct populations are present in the data set? How many people belong to each population?

Ex. 8 — Is the case/control and quantitative trait disbalance between populations?

Ex. 9 — For the quantitative trait, what method corrects best for stratification (in terms of minimal residual inflation)?

Ex. 10 — What is the strongest SNP associated with trait `quat`? What model (method and covariates used) gives best results? Is the finding GW-significant?

Ex. 11 — What is the strongest SNP associated with trait `bint`? What model (method and covariates used) gives best results? Is the finding GW-significant?

7.2 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

$$Y = \mu + G + e \tag{7.1}$$

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:

$$\begin{aligned} L(\mu, \sigma^2_G, \sigma^2_e) = & -\frac{1}{2} \cdot log_e \left|(\Phi \cdot \sigma^2_G + I \cdot \sigma^2_e)\right| \\ & + (Y - \mu)^T \cdot (\Phi \cdot \sigma^2_G + I \cdot \sigma^2_e)^{-1} \cdot (Y - \mu) \end{aligned} \tag{7.2}$$

where μ is intercept, σ^2_G is the proportion of variance explained by the polygenic component, and σ^2_e 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

$$\begin{aligned} L(\mu, \sigma^2_G, \sigma^2_e, \beta_1, \beta_2, ...) = & -\frac{1}{2} \cdot log_e \left|(\Phi \cdot \sigma^2_G + I \cdot \sigma^2_e)\right| \\ & + \left(Y - (\mu + \sum_j \beta_j \cdot C_j)\right)^T \cdot \left(\Phi \cdot \sigma^2_G + I \cdot \sigma^2_e\right)^{-1} \\ & \cdot \left(Y - (\mu + \sum_j \beta_j \cdot C_j)\right) \end{aligned} \tag{7.3}$$

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 (7.3). 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 (7.1) and likelihood (7.2) 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 (7.1). 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.

7.3 Example GWA analysis using family-based data

DATAPROBLEM In this section, we will explore small data set (notWorkingSexprnids(erfs) people, notWorkingSexprnsnps(erfs) SNPs). Let us load and explore it: **DATAPROBLEM trying to keep things working technically - the interpretation is not sane anymore**

```
> #load("RData/erfsmall.RData")
> data(ge03d2.clean)
> erfs <- ge03d2.clean[1:100,]
> phdata(erfs)$qtbas <- phdata(erfs)$weight
> pkins <- matrix(rnorm(nids(erfs)^2,sd=0.01),ncol=nids(erfs),nrow=nids(erfs))
> ls()

[1] "erfs"         "ge03d2.clean" "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-package`'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

```
> nids(erfs)

[1] 100

> nsnps(erfs)

[1] 7374
```

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

```
> table(chromosome(erfs))
```

```
   1    2    3    X
3482 1927 1417  548
```

(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(gtdata(erfs))

$`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   336.000     1289.000    1275.000    1625.00 2849.000
Prop   0.046        0.175       0.173       0.22    0.386

$`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        1  21.000 182.000  7374
Prop        0        0   0.003   0.025     1

$`Distribution of proportion of successful genotypes (per person)`
     X<=0.9 0.9<X<=0.95 0.95<X<=0.98 0.98<X<=0.99 X>0.99
No        0           0            0        56.00  44.00
Prop      0           0            0         0.56   0.44

$`Distribution of proportion 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        0      28.000     1967.000     2710.000 2669.000
Prop      0       0.004        0.267        0.368    0.362

$`Mean heterozygosity for a SNP`
[1] 0.2565366

$`Standard deviation of the mean heterozygosity for a SNP`
[1] 0.1627121

$`Mean heterozygosity for a person`
[1] 0.2460929

$`Standard deviation of mean heterozygosity for a person`
[1] 0.01891617
```

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]

              [,1]         [,2]         [,3]         [,4]         [,5]
[1,]  4.398682e-06 -0.007443991  0.001027849 -0.001690993  0.003792633
```

```
[2,] -2.690560e-03  0.005686529 -0.002653584 -0.002288409 -0.002089908
[3,] -1.551598e-02  0.002559050 -0.005672204  0.000416812  0.012694795
[4,] -1.523381e-02 -0.004245080  0.013530972 -0.008887517 -0.005388854
[5,]  7.371455e-03  0.018060832 -0.000732690 -0.009430937 -0.009738570
```

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.
-3.444e-02 -6.713e-03  1.919e-05  5.319e-05  6.701e-03  4.329e-02
```

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 7.4A):

```
> 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]
```

```
              id4         id10          id25         id33        id35
id4   0.443821169 6.587000e+03  6.592000e+03 6601.0000000 6587.000000
id10  0.007417817 4.624733e-01  6.582000e+03 6592.0000000 6577.000000
id25 -0.005339590 3.822318e-03  4.783133e-01 6595.0000000 6583.000000
id33  0.001082992 1.240865e-02 -2.744355e-02    0.4946951 6590.000000
id35 -0.029822882 1.299173e-02  5.400605e-03    0.0161940    0.494195
```

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.
-0.074800 -0.020410 -0.008065 -0.005080  0.006761  0.154700
```

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 7.4B):

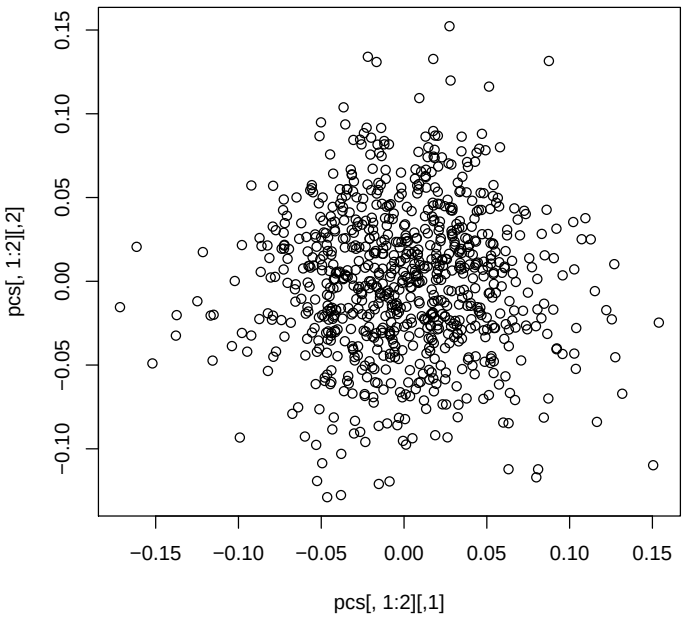

Figure 7.3: Population structure of '`strdat1`' data

```
> 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 7.5), and estimate correlation between the two with

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

```
[1] 0.01220139
```

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:

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

You can check the estimate of the inflation factor λ with

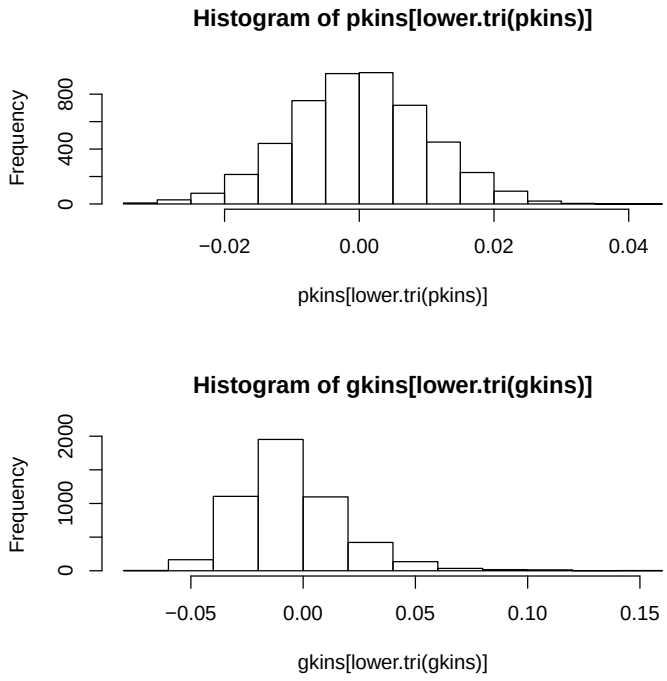

Figure 7.4: Distribution of the pedigree (upper histogram) and genomic (lower histogram) kinship coefficients for `erfs` data set.

```
> lambda(qts)$est

[1] 1.033775
```

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")

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2   N      effB  se_effB chi2.1df
rs5941875          1  3640023      +  T  A  97  19.68681 4.942356 15.86656
rs8994834          1  1845763      +  T  G  98  14.48396 3.898478 13.80335
rs4550412          1  1844249      +  A  G  99  15.14198 4.147116 13.33131
rs9253916          1  3655610      +  C  A 100  15.37248 4.237537 13.16015
rs9154666          1  3689697      +  G  C  99  14.77000 4.093217 13.02062
rs762401           1  3689720      +  T  C  99 -14.69329 4.090225 12.90457
rs7881009          2  7390224      -  C  A 100 -33.38444 9.454799 12.46762
rs3361134          3 11192711      +  G  A  99 -13.37212 3.801274 12.37491
rs6927636          1   385190      -  G  A  99  15.40149 4.421136 12.13551
rs1289801          1  4041666      +  G  T 100  26.26488 7.627108 11.85854
                 P1df      effAB     effBB chi2.2df         P2df        Pc1df
```

```
rs5941875 6.796909e-05  22.154991  24.12716 16.84336 0.0002200451 8.940686e-05
rs8994834 2.029744e-04  13.336702  29.93850 13.88632 0.0009652165 2.580954e-04
rs4550412 2.610112e-04  12.319392  35.27284 13.97540 0.0009231695 3.293272e-04
rs9253916 2.859663e-04  21.651670  18.72325 16.14362 0.0003122177 3.597997e-04
rs9154666 3.080797e-04  13.685849  31.22516 13.11489 0.0014195048 3.867340e-04
rs762401  3.277815e-04 -17.539309 -31.11841 13.00428 0.0015002246 4.106811e-04
rs7881009 4.140675e-04 -33.384438        NA 12.46762 0.0004140675 5.150733e-04
rs3361134 4.351428e-04  -6.364411 -26.42515 14.11220 0.0008621359 5.404643e-04
rs6927636 4.947085e-04  16.864623  27.51095 12.35393 0.0020767259 6.120307e-04
rs1289801 5.739765e-04  26.264881        NA 11.85854 0.0005739765 7.068666e-04
```

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)

  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

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

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2   N      effB  se_effB chi2.1df
rs5941875          1  3640023      +  T  A  97  19.68681 4.942356 15.86656
rs8994834          1  1845763      +  T  G  98  14.48396 3.898478 13.80335
rs4550412          1  1844249      +  A  G  99  15.14198 4.147116 13.33131
rs9253916          1  3655610      +  C  A 100  15.37248 4.237537 13.16015
rs9154666          1  3689697      +  G  C  99  14.77000 4.093217 13.02062
rs762401           1  3689720      +  T  C  99 -14.69329 4.090225 12.90457
rs7881009          2  7390224      -  C  A 100 -33.38444 9.454799 12.46762
rs3361134          3 11192711      +  G  A  99 -13.37212 3.801274 12.37491
rs6927636          1   385190      -  G  A  99  15.40149 4.421136 12.13551
rs1289801          1  4041666      +  G  T 100  26.26488 7.627108 11.85854
           P1df Pc1df      effAB     effBB chi2.2df P2df
rs5941875 0.130 0.185  22.154991  24.12716 16.84336   NA
rs8994834 0.395 0.485  13.336702  29.93850 13.88632   NA
rs4550412 0.490 0.575  12.319392  35.27284 13.97540   NA
rs9253916 0.515 0.615  21.651670  18.72325 16.14362   NA
rs9154666 0.545 0.645  13.685849  31.22516 13.11489   NA
rs762401  0.575 0.680 -17.539309 -31.11841 13.00428   NA
rs7881009 0.680 0.770 -33.384438        NA 12.46762   NA
rs3361134 0.695 0.775  -6.364411 -26.42515 14.11220   NA
rs6927636 0.755 0.820  16.864623  27.51095 12.35393   NA
rs1289801 0.800 0.875  26.264881        NA 11.85854   NA
```

(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 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] 747.5489

$estimate
[1]  92.3114671   0.4273438 663.0750850

$gradient
[1] -1.421085e-07 -7.105427e-08  1.136868e-07

$code
[1] 1

$iterations
[1] 7
```

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:

```
> lambda(mms)$est

[1] 1
```

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

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

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2   N      effB  se_effB chi2.1df
rs5941875          1  3640023      +  T  A  97  20.31359 5.201615 15.25096
```

```
rs1289801          1  4041666      +  G  T 100  29.17214 7.817672 13.92457
rs4550412          1  1844249      +  A  G  99  15.94462 4.311427 13.67686
rs8994834          1  1845763      +  T  G  98  14.75871 4.061652 13.20357
rs9253916          1  3655610      +  C  A 100  16.17211 4.596761 12.37740
rs6927636          1   385190      -  G  A  99  15.46655 4.437391 12.14875
rs6191668          1  3630425      -  G  A 100  17.23958 5.060666 11.60481
rs3361134          3 11192711      +  G  A  99 -13.05935 3.912352 11.14210
rs9154414          1  1836956      -  A  C 100  14.27818 4.307012 10.98990
rs7881009          2  7390224      -  C  A 100 -31.57798 9.543831 10.94771
                  P1df        Pc1df effAB effBB chi2.2df P2df
rs5941875 0.0000941288 0.0000941288    NA    NA        0   NA
rs1289801 0.0001902950 0.0001902950    NA    NA        0   NA
rs4550412 0.0002171135 0.0002171135    NA    NA        0   NA
rs8994834 0.0002794163 0.0002794163    NA    NA        0   NA
rs9253916 0.0004345614 0.0004345614    NA    NA        0   NA
rs6927636 0.0004912069 0.0004912069    NA    NA        0   NA
rs6191668 0.0006578149 0.0006578149    NA    NA        0   NA
rs3361134 0.0008439010 0.0008439010    NA    NA        0   NA
rs9154414 0.0009160997 0.0009160997    NA    NA        0   NA
rs7881009 0.0009371951 0.0009371951    NA    NA        0   NA
```

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$pgres,data=erfs,clam=FALSE)
> lambda(grs)$est

[1] 0.8998862
```

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")

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2   N       effB  se_effB  chi2.1df
rs5941875          1  3640023      +  T  A  97  11.049817 2.892067 14.598002
rs1289801          1  4041666      +  G  T 100  16.025348 4.475877 12.819130
rs4550412          1  1844249      +  A  G  99   8.433205 2.431798 12.026249
rs6927636          1   385190      -  G  A  99   8.772838 2.592604 11.450066
rs8994834          1  1845763      +  T  G  98   7.733678 2.293640 11.368989
rs7881009          2  7390224      -  C  A 100 -17.886196 5.548436 10.391897
rs9253916          1  3655610      +  C  A 100   7.931660 2.486748 10.173366
rs3361134          3 11192711      +  G  A  99  -7.058117 2.231245 10.006525
```

```
rs6191668          1  3630425      -  G  A 100   8.580199 2.757873  9.679351
rs9154414          1  1836956      -  A  C 100   7.501683 2.411571  9.676480
                 P1df      effAB      effBB chi2.2df         P2df        Pc1df
rs5941875 0.0001330556  12.468348  13.337066 15.54028 0.0004221544 0.0000563345
rs1289801 0.0003430932  16.025348         NA 12.81913 0.0003430932 0.0001604628
rs4550412 0.0005245653   7.599102  18.340676 12.18983 0.0022543047 0.0002564749
rs6927636 0.0007149146   9.564196  15.765123 11.63588 0.0029737318 0.0003610113
rs8994834 0.0007468052   7.076555  16.023283 11.44763 0.0032672281 0.0003788329
rs7881009 0.0012656955 -17.886196         NA 10.39190 0.0012656955 0.0006782175
rs9253916 0.0014248395  11.762204   8.529627 13.39739 0.0012325173 0.0007729162
rs3361134 0.0015598655  -3.716155 -13.964066 11.15333 0.0037851766 0.0008541251
rs6191668 0.0018635041  11.907381   3.293695 13.63234 0.0010959101 0.0010393086
rs9154414 0.0018664189   6.073566  17.517734 10.15451 0.0062370161 0.0010411021
```

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)

  |
  |                                                                      |   0%
  |
  |======================================================================| 100%

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

Summary for top 10 results, sorted by Pc1df
          Chromosome Position Strand A1 A2   N       effB  se_effB  chi2.1df
rs5941875          1  3640023      +  T  A  97  11.049817 2.892067 14.598002
rs1289801          1  4041666      +  G  T 100  16.025348 4.475877 12.819130
rs4550412          1  1844249      +  A  G  99   8.433205 2.431798 12.026249
rs6927636          1   385190      -  G  A  99   8.772838 2.592604 11.450066
rs8994834          1  1845763      +  T  G  98   7.733678 2.293640 11.368989
rs7881009          2  7390224      -  C  A 100 -17.886196 5.548436 10.391897
rs9253916          1  3655610      +  C  A 100   7.931660 2.486748 10.173366
rs3361134          3 11192711      +  G  A  99  -7.058117 2.231245 10.006525
rs9154414          1  1836956      -  A  C 100   7.501683 2.411571  9.676480
rs6191668          1  3630425      -  G  A 100   8.580199 2.757873  9.679351
           P1df Pc1df      effAB      effBB chi2.2df P2df
rs5941875 0.310 0.130  12.468348  13.337066 15.54028   NA
rs1289801 0.650 0.370  16.025348         NA 12.81913   NA
rs4550412 0.805 0.530   7.599102  18.340676 12.18983   NA
rs6927636 0.885 0.690   9.564196  15.765123 11.63588   NA
rs8994834 0.890 0.720   7.076555  16.023283 11.44763   NA
rs7881009 0.975 0.885 -17.886196         NA 10.39190   NA
rs9253916 0.980 0.900  11.762204   8.529627 13.39739   NA
rs3361134 0.985 0.910  -3.716155 -13.964066 11.15333   NA
rs9154414 1.000 0.935   6.073566  17.517734 10.15451   NA
rs6191668 1.000 0.935  11.907381   3.293695 13.63234   NA
```

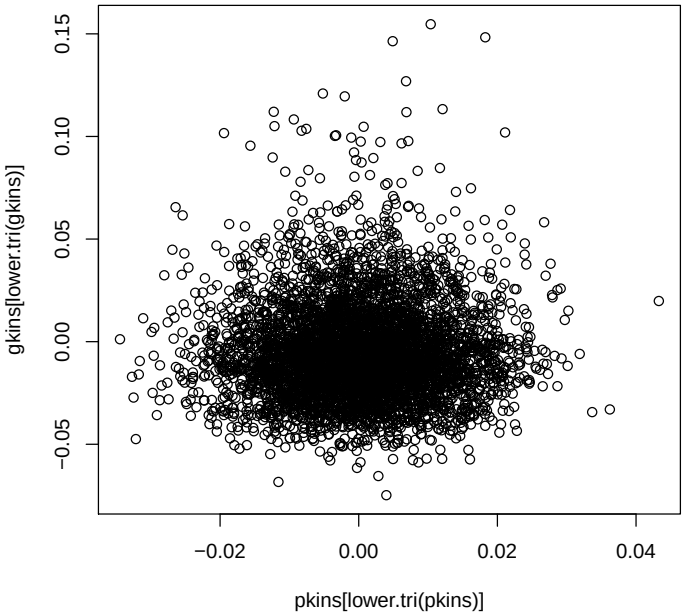

Figure 7.5: Scatter-plot relating pedigree and genomic kinships for `erfs` data set.

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 7.6):

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

7.4 Exercise: analysis of family data

Exercise 1 *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 2 *Describe the trait "qt". Can you detect significant outliers at visual inspection? Is trait distributed normally? What are significant covariates?*

Exercise 3 *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 4 *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 5 *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 6 *Repeat the last "best" analysis using pedigree kinship. How your results change?*

Exercise 7 *If you have any time left – repeat analysis using "qt" trait. This one is much more fun, but also more laborous to analyse.*

7.5 Answers to exercises

Answer (Ex. 1) — Basically, all covariates are significantly associated with the traits:

```
> summary(lm(quat~sex+age+age2,data=phdata(strdat)))

Call:
lm(formula = quat ~ sex + age + age2, data = phdata(strdat))

Residuals:
     Min       1Q   Median       3Q      Max
-21.3585  -4.9637   0.0959   4.7896  22.4212

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 167.935224   3.139965  53.483   <2e-16 ***
sex          12.102771   0.505588  23.938   <2e-16 ***
age          -0.025454   0.127427  -0.200    0.842
age2         -0.001697   0.001247  -1.361    0.174
---
Signif. codes:  0 âÃŸ***âÃŹ 0.001 âÃŸ**âÃŹ 0.01 âÃŸ*âÃŹ 0.05 âÃŸ.âÃŹ 0.1 âÃŸ âÃŹ 1
```

```
Residual standard error: 7.222 on 816 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.4517,        Adjusted R-squared:  0.4497
F-statistic:   224 on 3 and 816 DF,  p-value: < 2.2e-16
```

```
> summary(glm(bint~sex+age+age2,data=phdata(strdat),family=binomial))
```

```
Call:
glm(formula = bint ~ sex + age + age2, family = binomial, data = phdata(strdat))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3318  -1.1169  -0.8151   1.1907   1.5775

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.1234742  0.9295943  -3.360 0.000779 ***
sex          0.4913847  0.1424384   3.450 0.000561 ***
age          0.0976423  0.0373181   2.616 0.008884 **
age2        -0.0007977  0.0003623  -2.202 0.027697 *
---
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: 1137.4  on 821  degrees of freedom
Residual deviance: 1110.6  on 818  degrees of freedom
AIC: 1118.6

Number of Fisher Scoring iterations: 4
```

Answer (Ex. 2) — How many SNPs and IDs are presented in the data set?

```
> nsnps(strdat)
[1] 7374
> nids(strdat)
[1] 822
```

Answer (Ex. 3) — Let us perform the QC:

```
> qc <- check.marker(strdat,call=0.98,perid.call=0.98,p.level=0)
Excluding people/markers with extremely low call rate...
7374 markers and 822 people in total
0 people excluded because of call rate < 0.1
0 markers excluded because of call rate < 0.1
Passed: 7374 markers and 822 people

Running sex chromosome checks...
0 heterozygous X-linked male genotypes found
```

```
0 X-linked markers are likely to be autosomal (odds > 1000 )
0 male are likely to be female (odds > 1000 )
0 female are likely to be male (odds > 1000 )
0 people have intermediate X-chromosome inbreeding (0.5 > F > 0.5)
If these people/markers are removed, 0 heterozygous male genotypes are left
Passed: 7374 markers and 822 people

no X/Y/mtDNA-errors to fix


RUN 1
7374 markers and 822 people in total
0 (0%) markers excluded as having low (<0.3041363%) minor allele frequency
32 (0.4339571%) markers excluded because of low (<98%) call rate
0 (0%) markers excluded because they are out of HWE (P <0)
0 (0%) people excluded because of low (<98%) call rate
Mean autosomal HET is 0.2564845 (s.e. 0.01498989)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7911611 (s.e. 0.01163827), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 7342 (99.56604%) markers passed all criteria
In total, 822 (100%) people passed all criteria

RUN 2
7342 markers and 822 people in total
0 (0%) markers excluded as having low (<0.3041363%) minor allele frequency
0 (0%) markers excluded because of low (<98%) call rate
0 (0%) markers excluded because they are out of HWE (P <0)
0 (0%) people excluded because of low (<98%) call rate
Mean autosomal HET is 0.2564845 (s.e. 0.01498989)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7873167 (s.e. 0.01186485), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 7342 (100%) markers passed all criteria
In total, 822 (100%) people passed all criteria
```

```
> strdat1 <- strdat[qc$idok,qc$snpok]
```

The number of IDs and SNPs passing are

```
> nsnps(strdat1)

[1] 7342

> nids(strdat1)

[1] 822
```

Answer (Ex. 4) — Not really:

```
> descriptives.marker(strdat1)[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          2    5.000  64.000 316.000  7342
Prop        0    0.001   0.009   0.043     1
```

Answer (Ex. 5) — Yes:

```
> qts.q <- qtscore(quat~sex+age+age2,strdat1)
> lambda(qts.q)
$estimate
[1] 1.29321

$se
[1] 0.005757743
```

Answer (Ex. 6) — Yes:

```
> qts.b <- qtscore(bint~sex+age+age2,strdat1)
> lambda(qts.b)
$estimate
[1] 1.153616

$se
[1] 0.0011015
```

Answer (Ex. 7) — Two genetically distinct populations are present in the data set. These can be visualised with

```
> gkin <- ibs(strdat1,w="freq")
> dst <- as.dist(0.5-gkin)
> pcs <- cmdscale(dst,k=10)
> plot(pcs[,1:2])
```

(graph presented in figure 7.3)
The populations can be dustinguished with

```
> pop <- 1*(pcs[,1]>0)
```

and the number of people belonging to each can be computed with

```
> table(pop)
pop
  0   1
414 408
```

Answer (Ex. 8) — Yes, there is significant case/control disbalance:

```
> descriptives.trait(strdat1,by=pop)
       No(by.var=0)     Mean       SD No(by.var=1)     Mean       SD   Ptt
id              414       NA       NA          408       NA       NA    NA
sex             414    0.519    0.500          408    0.547    0.498 0.434
age             414   50.355   13.082          408   49.609   12.890 0.410
```

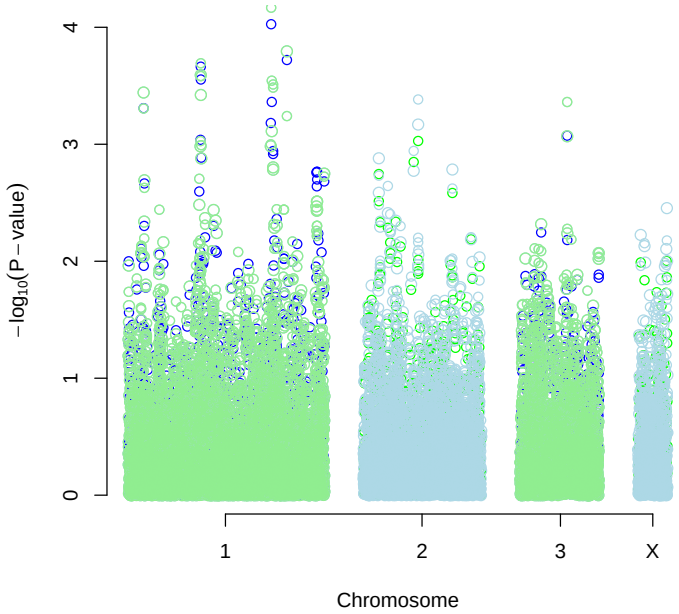

Figure 7.6: Comparison of FASTA (red), GRAMMAS (green), and plain GC (black).

```
dm2              414    0.505    0.501          408    0.444    0.497 0.079
height           414  168.399    9.952          406  168.755    9.518 0.601
weight           414   85.685   24.093          406   84.043   23.348 0.322
diet             414    0.043    0.204          408    0.047    0.211 0.831
bmi              414   30.198    8.285          406   29.421    7.574 0.161
quat             414  168.399    9.952          406  168.755    9.518 0.601
bint             414    0.505    0.501          408    0.444    0.497 0.079
age2             414 2706.340 1331.713          408 2626.745 1322.704 0.390
         Pkw Pexact
id        NA     NA
sex    0.434  0.443
age    0.313     NA
dm2    0.079  0.081
height 0.559     NA
weight 0.325     NA
diet   0.831  0.868
bmi    0.231     NA
quat   0.559     NA
bint   0.079  0.081
age2   0.313     NA
```

Answer (Ex. 9) — MMSCORE corrects best for stratification in terms of minimal residual inflation:

```
> sa <- qtscore(quat~sex+age+age2,strdat1,strat=pop)
> lambda(sa)$est
[1] 1.30266
> es <- egscore(quat~sex+age+age2,strdat1,kin=gkin)
> lambda(es)$est
[1] 1.332255
> pcs <- cmdscale(as.dist(0.5-gkin),k=10)
> pc <- qtscore(quat~sex+age+age2+pcs[,1]+pcs[,2]+pcs[,3],strdat1,strat=pop)
> lambda(pc)$est
[1] 1.303465
> h2an <- polygenic(quat~sex+age+age2,data=strdat1,kin=gkin)
> h2an$h2an
$minimum
[1] 4053.14

$estimate
[1] 168.709434625  12.159162310  -0.058149589  -0.001385047   0.100428761
[6]  51.746547785

$gradient
[1]  8.225762e-01 -8.163948e-03  4.391218e+00  1.454718e+04 -2.778261e-01
[6] -3.008137e-02

$code
[1] 2

$iterations
[1] 2
> mm <- mmscore(h2an,strdat1)
> lambda(mm)$est
[1] 1.121713
```

Answer (Ex. 10) — The best results are achieved with `mmscore`:

```
> summary(mm)
Summary for top 10 results, sorted by P1df
          Chromosome Position Strand A1 A2   N      effB   se_effB chi2.1df
rs3436694          2  8921418      -  C  G 812 -4.626188 0.6358782 52.92960
rs70099            2  8857747      +  C  A 810 -5.193788 0.7277973 50.92694
rs3074653          2  8915495      -  G  C 813 -3.456679 0.5281765 42.83116
rs1801282          2  8931192      +  C  T 809 -2.993759 0.5435064 30.34062
rs3175719          2  8950441      +  A  C 812 -2.652433 0.5556236 22.78911
rs6392986          2  8935924      +  A  T 810  1.745768 0.4232456 17.01328
```

```
rs7217010          1  2495320      +  C  A 812  4.399529 1.0719319 16.84527
rs4277955          2  7088955      -  C  T 806 -1.931248 0.4909742 15.47243
rs645280           2  7087627      +  C  A 815  1.649827 0.4470985 13.61666
rs1351516          2  8602074      +  G  A 815 -2.154476 0.5852874 13.55018
                  P1df        Pc1df effAB effBB chi2.2df P2df
rs3436694 3.457202e-13 6.454717e-12    NA    NA        0   NA
rs70099   9.586886e-13 1.605502e-11    NA    NA        0   NA
rs3074653 5.967390e-11 6.438807e-10    NA    NA        0   NA
rs1801282 3.624556e-08 1.984181e-07    NA    NA        0   NA
rs3175719 1.807868e-06 6.563749e-06    NA    NA        0   NA
rs6392986 3.711923e-05 9.839634e-05    NA    NA        0   NA
rs7217010 4.055425e-05 1.065218e-04    NA    NA        0   NA
rs4277955 8.371737e-05 2.040332e-04    NA    NA        0   NA
rs645280  2.241872e-04 4.937396e-04    NA    NA        0   NA
rs1351516 2.322688e-04 5.096838e-04    NA    NA        0   NA
```

Chapter 8

Imperfect knowledge about genotypes

This chapter is mostly copy-pasted from `ProbABEL-package` manual – user is encouraged to check it directly for the latest version.

8.1 Motivation

Many statistical and experimental techniques, such as imputations and high-throughput sequencing, generate 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 contrary, when dealing with imputed or high-throughput sequencing data, for many of the genomic loci we are quite uncertain about the genotypic status of the person. Instead of dealing with known genotypes we work with a probability distribution that is based on observed information, and we have estimates that true underlying genotype is either AA, AB or BB. The degree of confidence about the real status is measured with the 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 solutions that generate unbiased estimates of association parameters and takes the probability distribution into account is achieved by performing association analysis by means of regression of the outcome of interest onto estimated genotypic probabilities.

The `ProbABEL-package` package was designed to perform such regression in a fast, memory-efficient and consequently genome-wide feasible manner. Currently, `ProbABEL-package` implements linear, logistic regression, and Cox proportional hazards models. The corresponding analysis programs are called `palinear`, `palogist`, and `pacoxph`.

8.2 Input files

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

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

The dose/probability file may be supplied in filevector format in which case `ProbABEL-package` will operate much faster, and in low-RAM mode (approx. $\approx$ 128 MB). See the R libraries `GenABEL-package` and `DatABEL-package` on how to convert MACH and IMPUTE files to filevector format (functions: `mach2databel()` and `impute2databel()`, respectively).

8.2.1 SNP information file

In the simplest scenario, the 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", the average maximum posterior probability and "Rsq", the proportion of variance decrease after imputations).

Actually, for `ProbABEL-package`, 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/examples/test.mlinfo`)

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

8.2.2 Genomic predictor file

Again, in the simplest scenario this is an MLDOSE or MLPROB 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 (MLDOSE). In an MLPROB file, two columns for each SNP correspond to posterior probability that person has two ($P_{A_1A_1}$) or one ($P_{A_1A_2}$) copies of allele 1. The first "special" column is made of the sequential id, followed by an arrow followed by study ID (the one specified in the MACH input files). The second column contains the method keyword (e.g. "MLDOSE").

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 the file `ProbABEL/examples/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.

Therefore, 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.

The dose/probability file may be supplied in filevector format (`.fvi` and `.fvd` files) in which case `ProbABEL` will operate much faster, and in low-RAM mode (approx. 128 MB). On the command line simply specify the `.fvi` file as argument for the `-dose` option (cf. section 8.3 for more information on the options accepted by `ProbABEL`). See the R libraries GenABEL and DatABEL on how to convert MACH and IMPUTE files to filevector format (functions: `mach2databel()` and `impute2databel()`, respectively).

8.2.3 Phenotypic file

The 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 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`; an example is given in the `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 an analysis which will consider -999.9 as indeed a true measurements of the trait/covariates.

In the 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/examples/height.txt`)

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

Clearly, you can for example include `sex` $\times$ `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 couple of example lines of a phenotypic information file designed for logistic regression analysis follow here (also to be found in `ProbABEL/examples/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 $\text{chd} \sim \mu + \text{sex} + \text{age} + \text{othercov}$.

In case of the 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 an event occurred and zero if censored). Columns starting from four (inclusive) specify covariates to be included into the analysis. An example few lines of a phenotypic information file designed for the Cox proportional hazards model analysis follow here (also to be found in `ProbABEL/examples/coxph_data.txt`)

You can see that for the first ten people, the event occurs for three of them, while for the other seven there is no event during the follow-up time, as indicated by the "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.

8.2.4 Optional map file

If you would like 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 (example can be found in `ProbABEL/examples/test.map`)

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.

8.3 Running an analysis

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

There are in total 11 command line options you can specify to the `ProbABEL-package` analysis functions `palinear` or `palogist`. 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/MLPROB) 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.txt)
--skipd       : [optional] how many columns to skip in predictor
    (dose/prob) file (default 2)
--ntraits     : [optional] how many traits are analysed (default 1)
--ngpreds     : [optional] how many predictor columns per marker
    (default 1 = MLDOSE; else use 2 for MLPROB)
--separat     : [optional] character to separate fields (default is space)
--score       : use score test
--no-head     : do not report header line
--allcov      : report estimates for all covariates (large outputs!)
--interaction : which covariate to use for interaction with SNP
  (default is no ineraction, 0)
```

```
--mmscore     : score test for association between a trait and genetic
    polymorphism, in samples of related individuals
--robust      : report robust (aka sandwich, aka Hubert-White) standard
    errors
--help        : print help
```

8.3.1 Basic analysis options

However, for a simple run only three options are mandatory, which specify the necessary files needed to run the regression analysis.

These options are `-dose` (or `-d`), specifying the genomic predictor/MLDOSE file described in sub-section 8.2.2; `-pheno` (or `-p`), specifying the phenotypic data file described in sub-section 8.2.3; and `-info` (or `-i`), specifying the SNP information file described in sub-section 8.2.1.

If you change to the `ProbABEL/examples` directory you can run an analysis of height by running

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

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

The analysis of a binary trait (e.g. chd) can be run with

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

To run a Cox proportional hazards model, try

```
user@server:~/ProbABEL/examples/$ ../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 the analysis options.

To run an analysis with MLPROB files, you need specify the MLPROB file with the `-d` option and also specify that there are two genetic predictors per SNP, e.g. you can run linear model with

```
user@server:~/ProbABEL/examples/$ ../bin/palinear -p height.txt
 -d test.mlprob -i test.mlinfo
 --ngpreds=2
```

8.3.2 Advanced analysis options

The option `-interaction` allows you to include interaction between SNPs and any covariate. If for example your model is

$$\text{trait} \sim \text{sex} + \text{age} + \text{SNP},$$

running the program with the option `-interaction=2` will model

$$\text{trait} \sim \text{sex} + \text{age} + \text{SNP} + \text{age} \times \text{SNP}.$$

The option `-robust` allows you to compute so-called "robust" (a.k.a. "sandwich", a.k.a. Hubert-White) standard errors (cf. section 8.7 "Methodology" for details).

With the option `-mmscore` a score test for association between a trait and genetic polymorphisms in samples of related individuals is performed. A file with the inverse of the variance-covarince matrix goes as input parameter with that option, e.g. `-mmscore <filename>`. The file has to contain the first column with id names exactly like in phenotype file, BUT OMITTING people with no measured phenotype. The rest is a matrix. The phenotype file in case of using the `-mmscore` argument may contain any amount of covariates (this is different from previous versions). The first column contains id names, the second the trait. The others are covariates.

An example of how a polygenic object estimated by `GenABEL-package` can be used with ProbABEL is provided in `ProbABEL/examples/mmscore.R`

Though technically `-mmscore` allows for inclusion of multiple covariates, these should be kept to minimum as this is a score test. We suggest that any covariates explaining an essential proportion of variance should be fit as part of `GenABEL-package`'s `polygenic` procedure.

8.3.3 Running multiple analyses at once: `probabel.pl`

The Perl script `bin/probabel.pl_example` represents a handy wraper for `ProbABEL-package` functions. To start using it the configuration file `bin/probabel_config.cfg_example` needs to be edited. The configuration file consists of five columns. Each column except the first is a pattern for files produced by `MACH` (imputation software). The column named "cohort" is an identifying name of a population ("ERGO" in this example), the column "mlinfo_path" is the full path to mlinfo files, including a pattern where the chromosome number has been replaced by `_._chr_._`. The columns "mldose_path", "mlprobe_path" and "legend_path" are paths and patterns for "mldose", "mlprob" and "legend" files, respectively. These also need to include the pattern for the chromosome as used in the column for the "mlinfo" files. Probably you also have to change the variable `$config` in the script to point to the full path of the configuration file and the variable `@anprog` to point full path to the `ProbABEL-package` scripts.

8.4 Output file format

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

Here, only the first three lines of output have been shown. Note that lines starting with `+>` are actually the ones continuing the previous line – they have just been wrapped so we can see these long lines.

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

In 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 alarmed if the number here is

exactly the number of people in the file – this may indicate you did not code missing values according to `ProbABEL-package` format ('NA', 'NaN', or 'N').

The next column, nine ("Mean_predictor_allele"), gives the estimated frequency of the predictor allele (`A1`) in subjects with complete phenotypic data.

If the `-chrom` option was used, in the next column you will find the value specified by this option. If `-map` option was used, in the subsequent column you will find map location taken from the map-file. The subsequent columns provide coefficients of regression of the phenotype onto genotype, corresponding standard errors, and Wald χ^2 test value.

8.5 Preparing input files

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

8.6 Memory use and performance

Maximum likelihood regression is implemented in `ProbABEL-package`. With 6,000 people and 2.5 millions SNPs, a genome-wide scan is completed in less that an hour for a linear model with 1-2 covariates and overnight for logistic regression or the Cox proportional hazards model (figures for a PC bought back in 2007).

Memory may be an issue with `ProbABEL-package` if you use MACH text dose/probability files, e.g. for large chromosomes, such as chromosome one consumed up to 5 GB of RAM with 6,000 people.

We suggest that dose/probability file is to be supplied in filevector format in which case `ProbABEL-package` will operate about 2-3 times faster, and in low-RAM mode (approx. 128 MB). See the R libraries `GenABEL-package` and `DatABEL-package` on how to convert MACH and IMPUTE files to filevector format (functions: `mach2databel()` and `impute2databel()`, respectively).

When '–mmscore' option is used, the analysis may take quite some time.

8.7 Methodology

8.7.1 Analysis of population-based data

Linear regression assuming normal distribution

Standard linear regression theory is used to estimate coefficients of regression and their standard errors. We assume a linear model with expectation

$$E[\mathbf{Y}] = \mathbf{X}\boldsymbol{\beta} \tag{8.1}$$

and variance-covariance matrix

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

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

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

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{8.2}$$

and the MLE of the residual variance is

$$\hat{\sigma}^2 = \frac{(\mathbf{Y} - \mathbf{X}\hat{\beta})^T(\mathbf{Y} - \mathbf{X}\hat{\beta})}{N - r_X}, \tag{8.3}$$

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

The variance-covariance matrix for the parameter estimates under alternative hypothesis can be computed as

$$\mathbf{var}_{\hat{\beta}} = \hat{\sigma}^2(\mathbf{X}^T\mathbf{X})^{-1}. \tag{8.4}$$

For the j-the element $\hat{\beta}(j)$ of the vector of estimates the standard error under the alternative hypothesis is given by the square root of the corresponding diagonal element of the above matrix, $\mathbf{var}_{\hat{\beta}}(jj)$, and the Wald test can be computed with

$$T^2(j) = \frac{\hat{\beta}(j)^2}{\mathbf{var}_{\hat{\beta}}(jj)},$$

which asymptotically follows the χ^2 distribution with one degree of freedom under the null hypothesis.

When testing significance for more than one parameter simultaneously, several alternatives are available. Let us first partition the vector of parameters into two components, $\beta = (\beta_g, \beta_x)$, and our interest is testing the parameters contained in β_g (SNP effects), while β_x (e.g. effects of sex, age, etc.) are considered nuisance parameters. Let us define the vector of the parameters of interest which are fixed to certain values under the null hypothesis as $\beta_{g,0}$.

Firstly, the likelihood ratio test can be obtained with

$$LRT = 2\left(\text{logLik}(\hat{\beta}_g, \hat{\beta}_x) - \text{logLik}(\beta_{g,0}, \hat{\beta}_x)\right),$$

which under the null hypothesis is asymptotically distributed as χ^2 with the number of degrees of freedom equal to the number of parameters specified by β_g. Assuming the normal distribution, the log-likelihood of a model specified by the vector of parameters β and residual variance σ^2 can be computed as

$$\text{logLik}(\beta, \sigma^2) = -\frac{1}{2}\left(N \cdot \log_e \sigma^2 + (\mathbf{Y} - \beta\mathbf{X})^T(\mathbf{I}/\sigma^2)(\mathbf{Y} - \beta\mathbf{X})\right).$$

Secondly, the Wald test can be used; for that the inverse variance-covariance matrix of $\hat{\beta}_g$ should be computed as

$$\mathbf{var}^{-1}_{\hat{\beta}_g} = \mathbf{var}^{-1}_{\hat{\beta}}(g,g) - \mathbf{var}^{-1}_{\hat{\beta}}(g,x)\left(\mathbf{var}^{-1}_{\hat{\beta}}(x,x)\right)^{-1}\mathbf{var}^{-1}_{\hat{\beta}}(x,g),$$

where $\mathbf{var}^{-1}_{\hat{\beta}}(a,b)$ correspond to sub-matrices of the inverse of the variance-covariance matrix of $\hat{\beta}$, involving either only parameters of interest (g,g), nuisance parameters (x,x) or combination of these (x,g), (g,x).

The Wald test statistics is then computed as

$$W^2 = (\hat{\beta}_g - \beta_{g,0})^T \ \mathbf{var}_{\hat{\beta}_g}^{-1}(\hat{\beta}_g - \beta_{g,0}),$$

which asymptotically follows the χ^2 distribution with the number of degrees of freedom equal to the number of parameters specified by β_g. The Wald test generally is computationally easier than the LRT, because it avoids estimation of the model specified by the parameter's vector $(\beta_{g,0}, \hat{\beta}_x)$.

Lastly, similar to the Wald test, the score test can be performed by use of $\mathbf{var}_{(\beta_{g,0},\hat{\beta}_x)}$ instead of $\mathbf{var}_{\hat{\beta}}$.

Logistic regression

For logistic regression, the procedure to obtain parameters estimates, their variance-covariance matrix, and tests are similar to these outlined above with several modifications.

The expectation of the binary trait is defined as the expected probability of the event as defined by the logistic function

$$E[\mathbf{Y}] = \pi = \frac{1}{1 + e^{-(\mathbf{X}\beta)}}.$$

The estimates of the parameters are obtained not in one step, as is the case of the linear model, but using an iterative procedure (iteratively re-weighted least squares). This procedure is not described here for the sake of brevity.

The log-likelihood of the data is computed using the binomial probability formula:

$$\text{logLik}(\beta) = \mathbf{Y}^T \log_e \pi + (\mathbf{1} - \mathbf{Y})^T \log_e(\mathbf{1} - \pi),$$

where $\log_e \pi$ is a vector obtained by taking the natural logarithm of every value contained in the vector π.

Robust variance-covariance matrix of parameter estimates

For a linear model, these are computed using formula

$$\mathbf{var}_r = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{R}\mathbf{X})(\mathbf{X}^T\mathbf{X})^{-1},$$

where $\mathbf{R}$ is a diagonal matrix containing squares of residuals of $\mathbf{Y}$. The same formula may be used for "standard" analysis, in which case the elements of the $\mathbf{R}$ matrix are constant, namely mean residual sum of squares (the estimate of σ^2).

Similar to that, the robust matrix is computed for logistic regression with

$$\mathbf{var}_r = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{R}\mathbf{X})(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1},$$

where $\mathbf{1}$ is the vector of ones and $\mathbf{W}$ is the diagonal matrix of "weights" used in logistic regression.

Cox proportional hazards model

The implementation of the Cox proportional hazard model used in `ProbABEL-package` is entirely based on the code of `R` library `survival` developed by Thomas Lumley (function `coxfit2`), and is therefore not described here.

Many thanks to Thomas for making his code available under GNU GPL!

8.7.2 Analysis of pedigree data

The framework for analysis of pedigree data follows the two-step logic developed in the works of Aulchenko *et al.* (2007) and Chen and Abecasis (2007). General analysis model is a linear mixed model which defines the expectation of the trait as

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

identical to that defined for linear model (cf. section 8.1). To account for correlations between the phenotypes of relatives which may be induced by family relations the variance-covariance matrix is defined to be proportional to the linear combination of the identity matrix $\mathbf{I}$ and the relationship matrix $\mathbf{\Phi}$:

$$\mathbf{V}_{\sigma^2,h^2} = \sigma^2 \left(2h^2\mathbf{\Phi} + (1-h^2)\mathbf{I}\right),$$

where h^2 is the heritability of the trait. The relationship matrix $\mathbf{\Phi}$ is twice the matrix containing the coefficients of kinship between all pairs of individuals under consideration; its estimation is discussed in a separate section "8.7.2" (8.7.2).

Estimation of a model defined in such a way is possible by numerical maximization of the likelihood function, however, the estimation of this model for large pedigrees is laborious, and is not computationally feasible for hundreds of thousands to millions of SNPs to be tested in the context of GWAS, as we have demonstrated previously (Aulchenko *et al.*, 2007).

Two-step score test for association

A two-step score test approach is therefore used to decrease the computational burden. Let us first re-define the expectation of the trait by splitting the design matrix in two parts, the "base" part $\mathbf{X}_x$, which includes all terms not changing across all SNP models fit in GWAS (e.g. effects of sex, age, etc.), and the part including SNP information, $\mathbf{X_g}$:

$$E[\mathbf{Y}] = \mathbf{X}_x\beta_x + \mathbf{X}_g\beta_g.$$

Note that the latter design matrix may include not only the main SNP effect, but e.g. SNP by environment interaction terms.

At the first step, a linear mixed model not including SNP effects

$$E[\mathbf{Y}] = \mathbf{X}_x\beta_x$$

is fitted. The maximum likelihood estimates (MLEs) of the model parameters (regression coefficients for the fixed effects $\hat{\beta}_x$, the residual variance $\hat{\sigma}^2_x$ and the heritability $\hat{h}^2_x$) can be obtained by numerical maximization of the likelihood function

$$\text{logLik}(\beta_x, h^2, \sigma^2) = -\frac{1}{2}\left(\log_e|\mathbf{V}_{\sigma^2,h^2}| + (\mathbf{Y}-\beta_x\mathbf{X}_x)^T\mathbf{V}^{-1}_{\sigma^2,h^2}(\mathbf{Y}-\beta_x\mathbf{X}_x)\right),$$

where $\mathbf{V}^{-1}_{\sigma^2,h^2}$ is the inverse and $|\mathbf{V}_{\sigma^2,h^2}|$ is the determinant of the variance-covariance matrix.

At the second step, the unbiased estimates of the fixed effects of the terms involving SNP are obtained with

$$\hat{\beta}_g = (\mathbf{X}_g^T\mathbf{V}^{-1}_{\hat{\sigma}^2,\hat{h}^2}\mathbf{X}_g)^{-1}\mathbf{X}_g^T\mathbf{V}^{-1}_{\hat{\sigma}^2,\hat{h}^2}\mathbf{R}_{\hat{\beta}_x},$$

where $\mathbf{V}^{-1}_{\hat{\sigma}^2,\hat{h}^2}$ is the variance-covariance matrix at the point of the MLE estimates of $\hat{h}^2_x$ and $\hat{\sigma}^2_x$ and $\mathbf{R}_{\hat{\beta}_x} = \mathbf{Y} - \hat{\beta}_x \mathbf{X}_x$ is the vector of residuals obtained from the base regression model. Under the null model, the inverse variance-covariance matrix of the parameter's estimates is defined as

$$\mathbf{var}_{\hat{\beta}_g} = \hat{\sigma}^2_x (\mathbf{X}_g^T \mathbf{V}^{-1}_{\hat{\sigma}^2,\hat{h}^2} \mathbf{X}_g)^{-1}.$$

Thus the score test for joint significance of the terms involving SNP can be obtained with

$$T^2 = (\hat{\beta}_g - \beta_{g,0})^T \ \mathbf{var}^{-1}_{\hat{\beta}_g} (\hat{\beta}_g - \beta_{g,0}),$$

where $\beta_{g,0}$ are the values of parameters fixed under the null model. This test statistics under the null hypothesis asymptotically follows the χ^2 distribution with the number of degrees of freedom equal to the number of parameters tested. The significance of an individual j-the elements of the vector $\hat{\beta}_g$ can be tested with

$$T^2_j = \hat{\beta}^2_g(j)\, \mathbf{var}^{-1}_{\hat{\beta}_g}(jj),$$

where $\hat{\beta}^2_g(j)$ is the square of the j-th element of the vector of estimates $\hat{\beta}_g$, and $\mathbf{var}^{-1}_{\hat{\beta}_g}(jj)$ corresponds to the j-th diagonal element of $\mathbf{var}^{-1}_{\hat{\beta}_g}$. The latter statistics asymptotically follows χ^2_1.

Estimation of the kinship matrix

The relationship matrix $\mathbf{\Phi}$ used in estimation of the linear mixed model for pedigree data is twice the matrix containing the coefficients of kinship between all pairs of individuals under consideration. This coefficient is defined as the probability that two gametes randomly sampled from each member of the pair are identical-by-descent (IBD), that is they are copies of exactly the same ancestral allele. The expectation of kinship can be estimated from pedigree data using standard methods, for example the kinship for two outbred sibs is 1/4, for grandchild-grandparent is 1/8, etc. For an outbred person, the kinship coefficient is 1/2 – that is two gametes sampled from this person at random are IBD only if the same gamete is sampled. However, if the person is inbred, there is a chance that a maternal and paternal chromosomes are also IBD. The probability of this is characterized by kinship between individual's parents, which is defined as the individual's inbreeding coefficient, F. In this case, the kinship coefficient for the individual is $F + 1/2$. Similar logic applies to computation of the kinship coefficient for other types of pairs in inbred pedigrees.

The kinship matrix can be computed using the pedigree data using standard methods. However, in many cases, pedigree information may be absent, incomplete, or not reliable. Moreover, the estimates obtained using pedigree data reflect the expectation of the kinship, while the true realization of kinship may vary around this expectation. In presence of genomic data it may therefore be desirable to estimate the kinship coefficient from these, and not from pedigree. It can be demonstrated that unbiased and positive semi-definite estimator of the kinship matrix can be obtained (Astle and Balding, 2010; Amin *et al.*, 2007) by computing the kinship coefficients between individuals i and j with

$$\hat{K}_{ij} = \frac{1}{L} \sum_{l=1}^{L} \frac{(g_{l,i} - p_l)(g_{l,j} - p_l)}{p_l(1 - p_l)}$$

where L is the number of loci, p_l is the allelic frequency at l-th locus and $g_{l,j}$ is the genotype of j-th person at the l-th locus, coded as 0, 1/2, and 1, corresponding to the homozygous, heterozygous, and other type of homozygous genotype. The frequency is computed for the allele which, when homozygous, corresponds to the genotype coded as "1".

8.8 How to cite

If you used `ProbABEL-package` for your analysis please give a link to the `GenABEL project` home page

> http://www.genabel.org/

and cite the `ProbABEL-package` paper to give us some credit:

> Aulchenko YS, Struchalin MV, van Duijn CM.
> *ProbABEL package for genome-wide association analysis of imputed data.*
> BMC Bioinformatics. 2010, 11:134.

A proper reference may look like

> For the analysis of imputed data, we used the `ProbABEL-package` from the `GenABEL` suite of programs (Aulchenko *et al.*, 2010).

If you have used the Cox proportional hazard model, please mention the `R` package `survival` by Thomas Lumley. Additionally to the above citation, please tell that

> The Cox proportional hazards model implemented in `ProbABEL-package` makes use of the source code of the `R` package "`survival`" as implemented by T. Lumley.

Chapter 9

Analysis of imputed data: an example

In this chapter, you will perform an analysis of imputed data set. In this set of 120 individuals, 4500 SNPs are imputed based on information on 500 directly typed SNPs. You will first analyse 500 directly typed SNPs and then proceed to the analysis of imputed data. Finally, you will have a possibility to compare your results to the results of analysis in case all 5000 SNPs were directly typed.

9.1 Analysis of 500 directly typed SNPs

Load the `GenABEL-package` library and load the data set we will use:

```
> library(GenABEL)
> load("RData/ImputedDataAnalysis.RData")
> ls()

[1] "df500" "df5k"  "map5k" "old"   "rcT"
```

Here, `df500` contains the data including 500 directly typed SNPs, and `rcT` is the vector containing the value of the trait of interest:

```
> nids(df500)

[1] 120

> nsnps(df500)

[1] 500

> length(rcT)

[1] 120

> rcT[1:10]

 [1] -0.5560057  1.1798113  0.2721125  0.2174227 -0.8193852 -1.8739403
 [7]  0.9530623  0.9022811 -1.0906900  0.5575454
```

In all analysis that follow, do disregard the Genomic Control and GC-corrected results: as we will analyse a small region with strong association, the GC can not be applied.

Let us start with analysis of directly typed SNPs. For that, we will use `mlreg` function of `GenABEL-package`. This function implements ML-regression and Wald test of significance[1]. This will later on allow us direct comparison with the results of `ProbABEL-package`, which implements the same testing procedure.

We can run regression of `rcT` on SNPs region-wise using

```
> qts500 <- mlreg(rcT~1,df500)
> qts500[1:5,]
```

```
           Chromosome Position Strand A1 A2   N         effB   se_effB
rs6139074          20    11244      +  A  C 120  0.488853649 0.2079789
rs13043000         20    13288      +  G  T 120 -0.002890489 0.4394487
rs6037629          20    43093      +  T  G 120  0.504885204 0.4040614
rs6052070          20    44931      +  A  G 120  0.171428327 0.3037772
rs6116135          20    46930      +  G  A 120 -0.239668333 0.4235491
                chi2.1df      P1df     Pc1df effAB effBB chi2.2df P2df
rs6139074  5.524833e+00 0.0187484 0.2123725    NA    NA       NA   NA
rs13043000 4.326395e-05 0.9947519 0.9972156    NA    NA       NA   NA
rs6037629  1.561315e+00 0.2114728 0.5073673    NA    NA       NA   NA
rs6052070  3.184599e-01 0.5725346 0.7646316    NA    NA       NA   NA
rs6116135  3.201945e-01 0.5714908 0.7640104    NA    NA       NA   NA
```

The summary for the SNPs, which show most significant association can be produced with

```
> bestHits500 <- descriptives.scan(qts500,top=10)

Summary for top 10 results, sorted by P1df

> bestHits500
```

```
          Chromosome Position Strand A1 A2   N       effB   se_effB chi2.1df
rs6039167         20   846271      +  G  A 120 -5.9451984 0.7731855 59.12420
rs7261762         20   853448      +  A  G 120 -6.7335160 0.9748760 47.70732
rs511582          20   868752      +  A  G 120 -0.9872044 0.1883796 27.46290
rs967789          20  2876445      +  A  G 120 -0.9289317 0.1949364 22.70814
rs642758          20  2831146      +  A  G 120 -0.9194347 0.1993352 21.27521
rs676749          20  2974069      +  A  T 120 -0.8711208 0.1900255 21.01517
rs577116          20  2818285      +  G  A 120 -0.9291733 0.2032178 20.90595
rs570673          20  2819015      +  C  G 120 -0.8728347 0.1973862 19.55377
rs6112914         20  2048348      +  A  G 120 -0.6753674 0.1543434 19.14715
rs6106161         20  1995179      +  T  G 120 -0.6253343 0.1609140 15.10208
                 P1df        Pc1df effAB effBB chi2.2df P2df
rs6039167 1.480274e-14 4.512105e-05    NA    NA       NA   NA
rs7261762 4.948399e-12 2.477531e-04    NA    NA       NA   NA
rs511582  1.601368e-07 5.429559e-03    NA    NA       NA   NA
rs967789  1.885672e-06 1.146299e-02    NA    NA       NA   NA
```

[1] In general the score test may be preferred because it is faster and more robust.

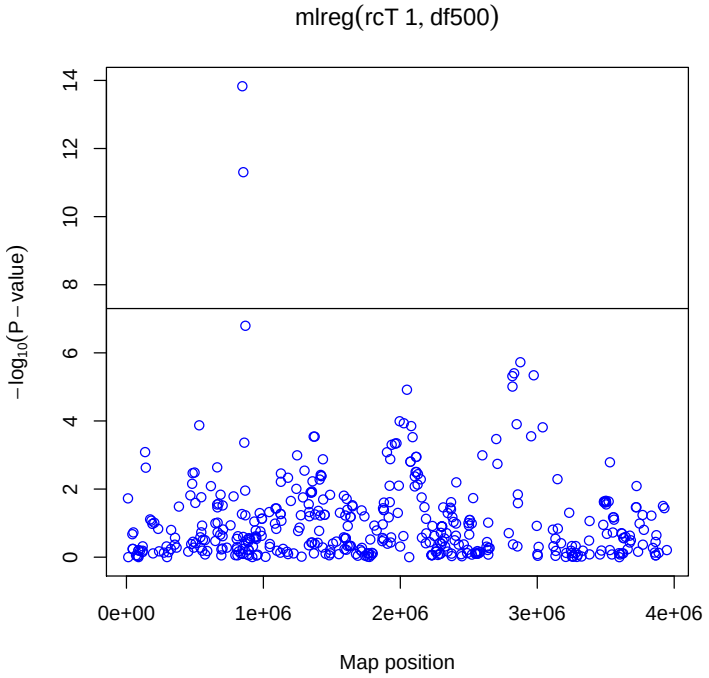

Figure 9.1: Manhattan plot for 500 directly typed SNPs

```
rs642758  3.978434e-06 1.439751e-02    NA    NA       NA   NA
rs676749  4.556610e-06 1.500792e-02    NA    NA       NA   NA
rs577116  4.823944e-06 1.527219e-02    NA    NA       NA   NA
rs570673  9.780761e-06 1.897119e-02    NA    NA       NA   NA
rs6112914 1.210181e-05 2.025583e-02    NA    NA       NA   NA
rs6106161 1.018509e-04 3.922563e-02    NA    NA       NA   NA
```

and the plot of the results with

```
> plot(qts500)
> abline(h=-log10(5e-8))
```

(see figure 9.1).

Finally, we can produce descriptive statistics for the SNPs, which demonstrated genome-wide significance:

```
> gwsSnps500 <- rownames(bestHits500)[bestHits500$P1df<=6e-8]
> gwsSnps500

[1] "rs6039167" "rs7261762"

> summary(gtdata(df500[,gwsSnps500 ]))

          Chromosome Position Strand A1 A2 NoMeasured CallRate         Q.2 P.11
rs6039167         20   846271      +  G  A        120        1 0.012500000  117
```

```
rs7261762         20   853448      +  A  G        120        1 0.008333333  118
          P.12 P.22 Pexact         Fmax      Plrt
rs6039167    3    0      1 -0.012658228 0.8454919
rs7261762    2    0      1 -0.008403361 0.8968501
```

Exercise 1.

It is known that rare variation in the presence of outliers can generate spurious associations. Do you believe this is a true association in this particular case? What you can do to check whether this is a true association or not?

9.2 Analysis of imputed data with `ProbABEL-package`

Here, you will analyse imputed data. In the `RData` directory, you will find the necessary files: `mach1.mldose.fvi` and `mach1.mldose.fvd` (these files represent `mldose` data produced by `mach1`, converted into `DatABEL-package` format[2]) and `mach1.out.mlinfo`, which contains information for the imputed SNPs generated in the process of imputations (such as 'Rsq', etc.).

We will start with producing a phenotypic data file for the use with `ProbABEL-package`:

```
> write.table(data.frame(id=idnames(df500), rcT=rcT),
+         file="rcT.PHE", quote=FALSE, row.names=FALSE)
```

next, try the command '`system("head rcT.PHE")`' to check the few first lines of the file.

At this moment, leave `R` (or, rather, start new console!), copy the mach-files to the working directory with

```
yourname@server> cp RData/mach1* .
```

and run `ProbABEL-package` analysis with

```
yourname@server> palinear --pheno rcT.PHE --info mach1.out.mlinfo /
                 --dose mach1.mldose.fvi
```

Do not forget to check that you start the analysis in right directory, i.e. all files (`rcT.PHE`, `mach1.out.mlinfo`, `mach1.mldose.fvi`, `mach1.mldose.fvd`) are present in the working directory (use the '`ls`' command from the console).

Now, you can return to `R` and load the analysis results:

```
> qtsPal <- read.table("regression_add.out.txt", head=T, strings=F)
> qtsPal[1:5,]
```

```
       name A1 A2  Freq1    MAF Quality    Rsq   n Mean_predictor_allele
1 rs4814683  T  G 0.4921 0.4921  0.7334 0.5650 120              0.492087
2 rs6076506  T  G 0.7723 0.2277  0.6511 0.2415 120              0.772271
3 rs6139074  C  A 0.3384 0.3384  0.9904 0.9803 120              0.338387
4 rs1418258  T  C 0.5009 0.4991  0.7208 0.5474 120              0.500917
5 rs7274499  C  A 0.9745 0.0255  0.9505 0.1011 120              0.974450
```

[2] using `mach2databel`

```
  beta_SNP_add sebeta_SNP_add   loglik
1     0.618307       0.269166 -114.266
2     0.722713       0.499781 -115.837
3     0.501472       0.210870 -114.082
4     0.636760       0.273391 -114.194
5     1.348270       2.085770 -116.679
```

As you see, there is not P-value produced in the `ProbABEL-package` output, and we need to compute it:

```
> qtsPal$Chisq <- (qtsPal$beta/qtsPal$sebeta)^2
> qtsPal$"P-value" <- pchisq(qtsPal$Chisq,1,low=F)
> qtsPal[1:5,]
```

```
       name A1 A2  Freq1    MAF Quality    Rsq   n Mean_predictor_allele
1 rs4814683  T  G 0.4921 0.4921  0.7334 0.5650 120              0.492087
2 rs6076506  T  G 0.7723 0.2277  0.6511 0.2415 120              0.772271
3 rs6139074  C  A 0.3384 0.3384  0.9904 0.9803 120              0.338387
4 rs1418258  T  C 0.5009 0.4991  0.7208 0.5474 120              0.500917
5 rs7274499  C  A 0.9745 0.0255  0.9505 0.1011 120              0.974450
  beta_SNP_add sebeta_SNP_add   loglik     Chisq    P-value
1     0.618307       0.269166 -114.266 5.2767671 0.02161184
2     0.722713       0.499781 -115.837 2.0910877 0.14816055
3     0.501472       0.210870 -114.082 5.6554059 0.01740165
4     0.636760       0.273391 -114.194 5.4247924 0.01985280
5     1.348270       2.085770 -116.679 0.4178505 0.51801156
```

Let us have a look at the top 20 associated SNPs:

```
> qtsPal[order(qtsPal$Chisq,decreasing=T)[1:20],]
```

```
           name A1 A2  Freq1    MAF Quality    Rsq   n Mean_predictor_allele
1009  rs6039167  G  A 0.9875 0.0125  0.9995 0.9800 120              0.987537
1022  rs7273309  C  T 0.9985 0.0015  0.9970 0.1896 120              0.998550
1026  rs7265788  A  G 0.9886 0.0114  0.9929 0.6884 120              0.988546
1025  rs8123328  G  T 0.9902 0.0098  0.9964 0.8126 120              0.990246
1024  rs7267882  G  A 0.9911 0.0089  0.9986 0.9230 120              0.991167
4009   rs566570  C  T 0.5307 0.4693  0.8814 0.8074 120              0.530712
1023  rs7261762  A  G 0.9913 0.0087  0.9991 0.9472 120              0.991346
2973  rs6035871  A  G 0.9387 0.0613  0.8962 0.2582 120              0.938696
2969  rs6047425  A  G 0.9388 0.0612  0.8964 0.2573 120              0.938767
3877  rs2325971  T  C 0.5811 0.4189  0.8526 0.7807 120              0.581042
1017   rs553378  G  A 0.9987 0.0013  0.9974 0.0005 120              0.998704
3880   rs873711  A  G 0.5888 0.4112  0.8597 0.7934 120              0.588750
3886  rs6037425  G  C 0.5871 0.4129  0.8651 0.8034 120              0.587125
3958  rs6037443  A  G 0.4975 0.4975  0.8918 0.8340 120              0.497446
4027  rs6076466  T  C 0.7040 0.2960  0.6570 0.3365 120              0.703938
3884  rs6051434  G  T 0.5186 0.4814  0.8427 0.7732 120              0.518621
1018   rs554362  A  G 0.6496 0.3504  0.5748 0.2481 120              0.649604
4032  rs2326056  C  A 0.7338 0.2662  0.6422 0.2403 120              0.733767
4029 rs11697448  G  A 0.7414 0.2586  0.6464 0.2370 120              0.741396
```

```
1560  rs6033304  A  C 0.8234 0.1766  0.7408 0.2813 120             0.823429
     beta_SNP_add sebeta_SNP_add    loglik    Chisq      P-value
1009      5.98851       0.781065  -92.6360 58.78455 1.759171e-14
1022     38.75330       5.101300  -93.0016 57.71061 3.036607e-14
1026      7.39562       0.977268  -93.1526 57.26929 3.800428e-14
1025      7.12447       0.984098  -94.8389 52.41172 4.500298e-13
1024      6.85041       0.977608  -96.0155 49.10250 2.429299e-12
4009      1.29428       0.185797  -96.2227 48.52651 3.258536e-12
1023      6.80755       0.977629  -96.2366 48.48792 3.323302e-12
2973      4.88622       0.721348  -97.1826 45.88348 1.254997e-11
2969      4.89489       0.722872  -97.1940 45.85251 1.275000e-11
3877      1.29728       0.192726  -97.3933 45.30925 1.682530e-11
1017    728.01600     108.683000  -97.5547 44.87026 2.105310e-11
3880      1.26975       0.192071  -97.9861 43.70316 3.821541e-11
3886      1.25994       0.190625  -97.9929 43.68577 3.855647e-11
3958      1.21385       0.184428  -98.1290 43.31879 4.650954e-11
4027      1.98273       0.339011 -101.6180 34.20579 4.958132e-09
3884      1.14070       0.199652 -102.2370 32.64341 1.107140e-08
1018      2.12435       0.376830 -102.5820 31.78053 1.726140e-08
4032      2.33945       0.427909 -103.3440 29.88992 4.572850e-08
4029      2.36276       0.435773 -103.5440 29.39805 5.893679e-08
1560      2.39452       0.451563 -104.0670 28.11902 1.140790e-07
```

and produce the plot of the results with

```
> plot(map5k,-log10(qtsPal[,"P-value"]))
> abline(h=-log10(5e-8))
```

(here, `map5k` contains map information for all 5000 SNPs).

To compare with the results obtained using 500 directly typed SNPs only, we can add the points with

```
> points(map(qts500),-log10(qts500[,"P1df"]),col="red",pch=19,cex=0.5)
```

(see figure 9.2).

Now, we can also check different characteristics of the SNPs, which are claimed to be significant in our analysis:

```
> gwsSnpsImp <- qtsPal$name[which(qtsPal[,"P-value"]<=5e-8)]
> gwsSnpsImp

 [1] "rs6039167" "rs553378"  "rs554362"  "rs7273309" "rs7261762" "rs7267882"
 [7] "rs8123328" "rs7265788" "rs6047425" "rs6035871" "rs2325971" "rs873711"
[13] "rs6051434" "rs6037425" "rs6037443" "rs566570"  "rs6076466" "rs2326056"

> mlinfo <- read.table("mach1.out.mlinfo",head=T,strings=F)
> mlinfo[mlinfo$SNP %in% gwsSnpsImp,]

           SNP Al1 Al2  Freq1    MAF Quality    Rsq
1009 rs6039167   G   A 0.9875 0.0125  0.9995 0.9800
1017  rs553378   G   A 0.9987 0.0013  0.9974 0.0005
1018  rs554362   A   G 0.6496 0.3504  0.5748 0.2481
```

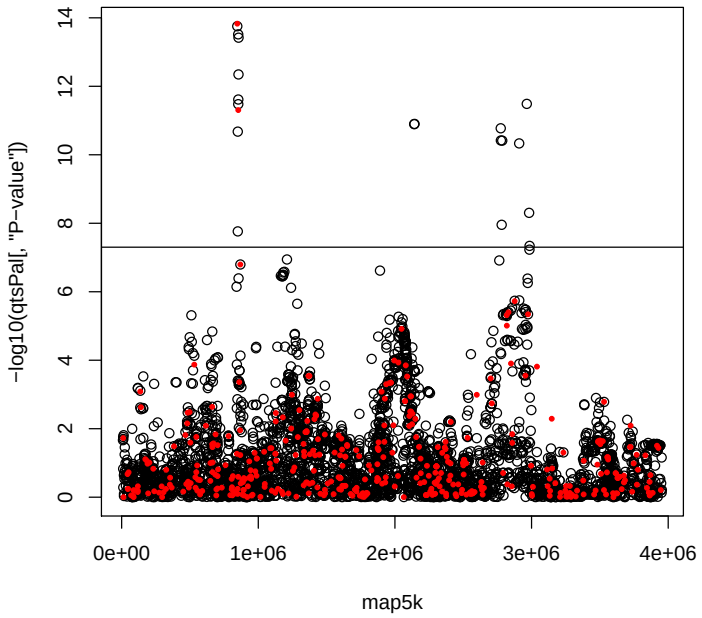

Figure 9.2: Manhattan plot for imputed (black empty circles) and 500 directly typed (red solid circles) SNPs

```
1022 rs7273309   C   T 0.9985 0.0015  0.9970 0.1896
1023 rs7261762   A   G 0.9913 0.0087  0.9991 0.9472
1024 rs7267882   G   A 0.9911 0.0089  0.9986 0.9230
1025 rs8123328   G   T 0.9902 0.0098  0.9964 0.8126
1026 rs7265788   A   G 0.9886 0.0114  0.9929 0.6884
2969 rs6047425   A   G 0.9388 0.0612  0.8964 0.2573
2973 rs6035871   A   G 0.9387 0.0613  0.8962 0.2582
3877 rs2325971   T   C 0.5811 0.4189  0.8526 0.7807
3880  rs873711   A   G 0.5888 0.4112  0.8597 0.7934
3884 rs6051434   G   T 0.5186 0.4814  0.8427 0.7732
3886 rs6037425   G   C 0.5871 0.4129  0.8651 0.8034
3958 rs6037443   A   G 0.4975 0.4975  0.8918 0.8340
4009  rs566570   C   T 0.5307 0.4693  0.8814 0.8074
4027 rs6076466   T   C 0.7040 0.2960  0.6570 0.3365
4032 rs2326056   C   A 0.7338 0.2662  0.6422 0.2403
```

Exercise 2.

Please classify the associations obtained into 'true' and 'false' (meaning 'I believe it' and 'I do not believe'). Of cause ultimate answer is replications. Still, the object `df5k` provides genotypes for all 5000 SNPs, directly typed! So,

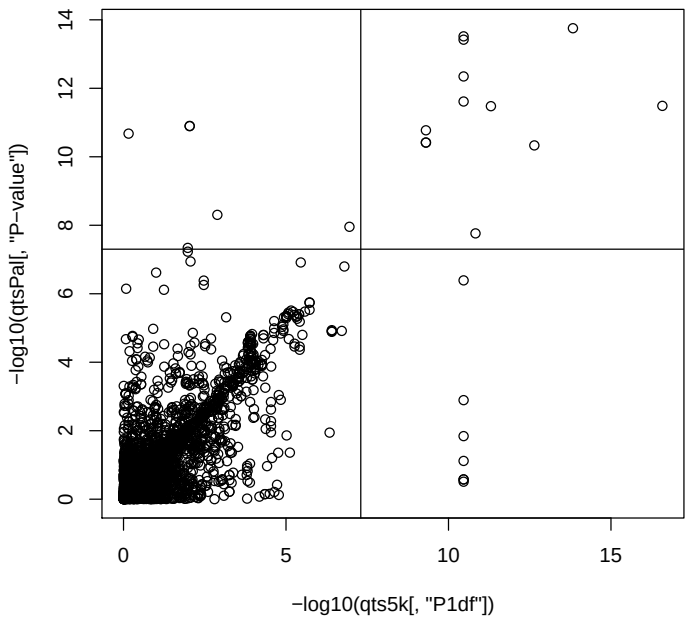

Figure 9.3: Cross-plot of the results from analysis of imputed and directly typed data (see answers to exercises).

you can run analysis, and cross-check which SNPs are confirmed as significant. Make a cross table: SNPs you thought were truly associated vs. SNPs indeed associated in directly typed data set.

9.3 Analysis of imputed data with `MixABEL-package`

In case you are interested in quantitative traits, `MixABEL-package` provides much faster and more flexible analysis tools, compared to the `ProbABEL-package`[3].

Here the above analysis is repeated using `MixABEL-package`.

Load necessary libraries:

```
> library(DatABEL)
> library(MixABEL)

MixABEL v 0.1-2 (November 23, 2012) loaded
```

Load the genotypic data:

```
> imp5k <- databel("mach1.mldose")
```

Run analysis and look up 10 top results:

[3]However, at the moment `MixABEL-package` misses functionality to analyse binary and time-till-event traits

```
> qtsImp <- GWFGLS(rcT~SNP,geno=imp5k,data=NULL)
> qtsImp[order(qtsImp[,"Chisq"],decreasing=T)[1:10],]
```

```
            n    Chisq df      P-value  beta_SNP    se_SNP mean_SNP
rs6039167 120 58.78462  1 1.759110e-14  5.988511 0.7810647 1.975075
rs7273309 120 57.71070  1 3.036458e-14 38.753300 5.1012957 1.997100
rs7265788 120 57.26926  1 3.800491e-14  7.395619 0.9772682 1.977092
rs8123328 120 52.41165  1 4.500458e-13  7.124468 0.9840984 1.980492
rs7267882 120 49.10249  1 2.429317e-12  6.850411 0.9776083 1.982333
rs566570  120 48.52652  1 3.258526e-12  1.294281 0.1857972 1.061425
rs7261762 120 48.48790  1 3.323324e-12  6.807552 0.9776294 1.982692
rs6035871 120 45.88351  1 1.254980e-11  4.886221 0.7213479 1.877392
rs6047425 120 45.85245  1 1.275033e-11  4.894889 0.7228723 1.877533
rs2325971 120 45.30891  1 1.682820e-11  1.297276 0.1927261 1.162083
```

Look up the information for SNPs with P-value less than $6e-8$:

```
> #plot(map5k,-log10(qtsImp[,"P-value"]))
> #abline(h=-log10(5e-8))
> gwsSnpsImp <- rownames(qtsImp)[which(qtsImp[,"P-value"]<=6e-8)]
> mlinfo <- read.table("mach1.out.mlinfo",head=T,strings=F)
> mlinfo[mlinfo$SNP %in% gwsSnpsImp,]
```

```
            SNP Al1 Al2  Freq1    MAF Quality    Rsq
1009  rs6039167   G   A 0.9875 0.0125  0.9995 0.9800
1017   rs553378   G   A 0.9987 0.0013  0.9974 0.0005
1018   rs554362   A   G 0.6496 0.3504  0.5748 0.2481
1022  rs7273309   C   T 0.9985 0.0015  0.9970 0.1896
1023  rs7261762   A   G 0.9913 0.0087  0.9991 0.9472
1024  rs7267882   G   A 0.9911 0.0089  0.9986 0.9230
1025  rs8123328   G   T 0.9902 0.0098  0.9964 0.8126
1026  rs7265788   A   G 0.9886 0.0114  0.9929 0.6884
2969  rs6047425   A   G 0.9388 0.0612  0.8964 0.2573
2973  rs6035871   A   G 0.9387 0.0613  0.8962 0.2582
3877  rs2325971   T   C 0.5811 0.4189  0.8526 0.7807
3880   rs873711   A   G 0.5888 0.4112  0.8597 0.7934
3884  rs6051434   G   T 0.5186 0.4814  0.8427 0.7732
3886  rs6037425   G   C 0.5871 0.4129  0.8651 0.8034
3958  rs6037443   A   G 0.4975 0.4975  0.8918 0.8340
4009   rs566570   C   T 0.5307 0.4693  0.8814 0.8074
4027  rs6076466   T   C 0.7040 0.2960  0.6570 0.3365
4029 rs11697448   G   A 0.7414 0.2586  0.6464 0.2370
4032  rs2326056   C   A 0.7338 0.2662  0.6422 0.2403
```

9.4 Answers to exercises

Answer (Ex. 1) — Firstly, you can check (by producing a cross-plot of genotype vs. phenotype) if association is indeed due to extreme phenotypic outliers. A related question is whether the distribution is skewed. Additionally, a

permutation-based test can help establishing correct p-value, taking into account the nature of the data in question.
However, to give an ultimate answer, a replication study is needed, in which these rare SNPs are to be typed in a large independent sample.

Answer (Ex. 2) — Here is the sequence of commands leading you to the answer:

```
> qts5k <- mlreg(rcT~1,df5k)
> bestHits5k <- descriptives.scan(qts5k,top=20)
Summary for top 20 results, sorted by P1df
> bestHits5k
```

	Chromosome	Position	Strand	A1	A2	N	effB	se_effB	chi2.1df
rs566570	20	2965113	+	T	C	120	1.341584	0.1585342	71.61267
rs6039167	20	846271	+	G	A	120	-5.945198	0.7731855	59.12420
rs6037443	20	2909408	+	G	A	120	1.194178	0.1628897	53.74655
rs7261762	20	853448	+	A	G	120	-6.733516	0.9748760	47.70732
rs554362	20	850002	+	A	G	120	-5.431215	0.8045925	45.56613
rs2104741	20	787070	+	G	A	120	-9.203756	1.3888630	43.91485
rs7273309	20	853154	+	C	T	120	-9.203756	1.3888630	43.91485
rs7267882	20	853785	+	G	A	120	-9.203756	1.3888630	43.91485
rs8123328	20	855045	+	G	T	120	-9.203756	1.3888630	43.91485
rs7265788	20	855426	+	A	G	120	-9.203756	1.3888630	43.91485
rs7263171	20	855655	+	C	T	120	-9.203756	1.3888630	43.91485
rs6110342	20	1458950	+	T	G	120	-9.203756	1.3888630	43.91485
rs6105340	20	1493635	+	G	A	120	-9.203756	1.3888630	43.91485
rs11905071	20	1557640	+	T	C	120	-9.203756	1.3888630	43.91485
rs6080013	20	1590861	+	G	A	120	-9.203756	1.3888630	43.91485
rs6074978	20	1595216	+	T	C	120	-9.203756	1.3888630	43.91485
rs2325971	20	2774477	+	T	C	120	-1.055991	0.1697536	38.69752
rs873711	20	2775468	+	A	G	120	-1.055991	0.1697536	38.69752
rs6037425	20	2785379	+	G	C	120	-1.055991	0.1697536	38.69752
rs6051434	20	2781237	+	T	G	120	0.938854	0.1769196	28.16076

	P1df	Pc1df	effAB	effBB	chi2.2df	P2df
rs566570	2.618695e-17	5.551496e-06	NA	NA	NA	NA
rs6039167	1.480274e-14	3.663781e-05	NA	NA	NA	NA
rs6037443	2.280947e-13	8.301445e-05	NA	NA	NA	NA
rs7261762	4.948399e-12	2.090409e-04	NA	NA	NA	NA
rs554362	1.475718e-11	2.904523e-04	NA	NA	NA	NA
rs2104741	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs7273309	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs7267882	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs8123328	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs7265788	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs7263171	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs6110342	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs6105340	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs11905071	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs6080013	3.429767e-11	3.745369e-04	NA	NA	NA	NA
rs6074978	3.429767e-11	3.745369e-04	NA	NA	NA	NA

```
rs2325971  4.948386e-10 8.395801e-04    NA    NA       NA   NA
rs873711   4.948386e-10 8.395801e-04    NA    NA       NA   NA
rs6037425  4.948386e-10 8.395801e-04    NA    NA       NA   NA
rs6051434  1.116452e-07 4.389549e-03    NA    NA       NA   NA
```

```
> plot(qts5k)
> abline(h=-log10(5e-8))
> gwsSnps5k <- rownames(bestHits5k)[bestHits5k$P1df<=6e-8]
> summary(gtdata(df5k[,gwsSnps5k ]))
```

```
           Chromosome Position Strand A1 A2 NoMeasured CallRate         Q.2
rs566570           20  2965113      +  T  C        120        1 0.495833333
rs6039167          20   846271      +  G  A        120        1 0.012500000
rs6037443          20  2909408      +  G  A        120        1 0.491666667
rs7261762          20   853448      +  A  G        120        1 0.008333333
rs554362           20   850002      +  A  G        120        1 0.012500000
rs2104741          20   787070      +  G  A        120        1 0.004166667
rs7273309          20   853154      +  C  T        120        1 0.004166667
rs7267882          20   853785      +  G  A        120        1 0.004166667
rs8123328          20   855045      +  G  T        120        1 0.004166667
rs7265788          20   855426      +  A  G        120        1 0.004166667
rs7263171          20   855655      +  C  T        120        1 0.004166667
rs6110342          20  1458950      +  T  G        120        1 0.004166667
rs6105340          20  1493635      +  G  A        120        1 0.004166667
rs11905071         20  1557640      +  T  C        120        1 0.004166667
rs6080013          20  1590861      +  G  A        120        1 0.004166667
rs6074978          20  1595216      +  T  C        120        1 0.004166667
rs2325971          20  2774477      +  T  C        120        1 0.470833333
rs873711           20  2775468      +  A  G        120        1 0.470833333
rs6037425          20  2785379      +  G  C        120        1 0.470833333
           P.11 P.12 P.22    Pexact         Fmax      Plrt
rs566570     33   55   32 0.3648758  0.083269672 0.3614010
rs6039167   117    3    0 1.0000000 -0.012658228 0.8454919
rs6037443    35   52   33 0.1467972  0.133092526 0.1442654
rs7261762   118    2    0 1.0000000 -0.008403361 0.8968501
rs554362    117    3    0 1.0000000 -0.012658228 0.8454919
rs2104741   119    1    0 1.0000000 -0.004184100 0.9484250
rs7273309   119    1    0 1.0000000 -0.004184100 0.9484250
rs7267882   119    1    0 1.0000000 -0.004184100 0.9484250
rs8123328   119    1    0 1.0000000 -0.004184100 0.9484250
rs7265788   119    1    0 1.0000000 -0.004184100 0.9484250
rs7263171   119    1    0 1.0000000 -0.004184100 0.9484250
rs6110342   119    1    0 1.0000000 -0.004184100 0.9484250
rs6105340   119    1    0 1.0000000 -0.004184100 0.9484250
rs11905071  119    1    0 1.0000000 -0.004184100 0.9484250
rs6080013   119    1    0 1.0000000 -0.004184100 0.9484250
rs6074978   119    1    0 1.0000000 -0.004184100 0.9484250
rs2325971    38   51   31 0.1032427  0.147097763 0.1065638
rs873711     38   51   31 0.1032427  0.147097763 0.1065638
rs6037425    38   51   31 0.1032427  0.147097763 0.1065638
```

```
> plot(-log10(qts5k[,"P1df"]),-log10(qtsPal[,"P-value"]))
```

```
> abline(h=-log10(5e-8))
> abline(v=-log10(5e-8))
> directNotImp <- gwsSnps5k[!(gwsSnps5k %in% gwsSnpsImp)]
> directNotImp
[1] "rs2104741"  "rs7263171"  "rs6110342"  "rs6105340"  "rs11905071"
[6] "rs6080013"  "rs6074978"
> imputeNotDir <- gwsSnpsImp[!(gwsSnpsImp %in% gwsSnps5k)]
> imputeNotDir
[1] "rs553378"  "rs6047425" "rs6035871" "rs6051434" "rs6076466" "rs2326056"
> inBoth <- gwsSnps5k[gwsSnps5k %in% gwsSnpsImp]
> inBoth
 [1] "rs566570"  "rs6039167" "rs6037443" "rs7261762" "rs554362"  "rs7273309"
 [7] "rs7267882" "rs8123328" "rs7265788" "rs2325971" "rs873711"  "rs6037425"
> summary(gtdata(df5k[,directNotImp]))
           Chromosome Position Strand A1 A2 NoMeasured CallRate         Q.2
rs2104741          20   787070      +  G  A        120        1 0.004166667
rs7263171          20   855655      +  C  T        120        1 0.004166667
rs6110342          20  1458950      +  T  G        120        1 0.004166667
rs6105340          20  1493635      +  G  A        120        1 0.004166667
rs11905071         20  1557640      +  T  C        120        1 0.004166667
rs6080013          20  1590861      +  G  A        120        1 0.004166667
rs6074978          20  1595216      +  T  C        120        1 0.004166667
           P.11 P.12 P.22 Pexact       Fmax     Plrt
rs2104741   119    1    0      1 -0.0041841 0.948425
rs7263171   119    1    0      1 -0.0041841 0.948425
rs6110342   119    1    0      1 -0.0041841 0.948425
rs6105340   119    1    0      1 -0.0041841 0.948425
rs11905071  119    1    0      1 -0.0041841 0.948425
rs6080013   119    1    0      1 -0.0041841 0.948425
rs6074978   119    1    0      1 -0.0041841 0.948425
> mlinfo[which(mlinfo$SNP %in% directNotImp),]
            SNP Al1 Al2  Freq1    MAF Quality    Rsq
871   rs2104741   G   A 0.9562 0.0438  0.9145 0.0679
1027  rs7263171   C   T 0.8639 0.1361  0.9246 0.7898
1978  rs6110342   T   G 0.9160 0.0840  0.8472 0.1967
2028  rs6105340   G   A 0.8790 0.1210  0.7849 0.2076
2064 rs11905071   T   C 0.8846 0.1154  0.7989 0.1890
2111  rs6080013   G   A 0.9577 0.0423  0.9342 0.3869
2118  rs6074978   T   C 0.9576 0.0424  0.9340 0.3851
> summary(gtdata(df5k[,imputeNotDir]))
           Chromosome Position Strand A1 A2 NoMeasured CallRate        Q.2 P.11
rs553378           20   849882      +  G  A        120        1 0.02500000  114
rs6047425          20  2141014      +  A  G        120        1 0.07083333  104
rs6035871          20  2143364      +  A  G        120        1 0.07083333  104
rs6051434          20  2781237      +  T  G        120        1 0.47500000   37
rs6076466          20  2982048      +  T  C        120        1 0.22916667   71
```

```
rs2326056         20  2985607      +  C  A        120        1 0.22500000   72
          P.12 P.22    Pexact          Fmax      Plrt
rs553378     6    0 1.0000000 -0.025641026 0.6948707
rs6047425   15    1 0.4561335  0.050382485 0.6092978
rs6035871   15    1 0.4561335  0.050382485 0.6092978
rs6051434   52   31 0.1473878  0.131161236 0.1502658
rs6076466   43    6 1.0000000 -0.014250614 0.8754486
rs2326056   42    6 1.0000000 -0.003584229 0.9686478
```

```
> mlinfo[which(mlinfo$SNP %in% imputeNotDir),]
```

```
           SNP Al1 Al2  Freq1    MAF Quality    Rsq
1017  rs553378   G   A 0.9987 0.0013  0.9974 0.0005
2969 rs6047425   A   G 0.9388 0.0612  0.8964 0.2573
2973 rs6035871   A   G 0.9387 0.0613  0.8962 0.2582
3884 rs6051434   G   T 0.5186 0.4814  0.8427 0.7732
4027 rs6076466   T   C 0.7040 0.2960  0.6570 0.3365
4032 rs2326056   C   A 0.7338 0.2662  0.6422 0.2403
```

```
> summary(gtdata(df5k[,inBoth]))
```

```
          Chromosome Position Strand A1 A2 NoMeasured CallRate         Q.2 P.11
rs566570          20  2965113      +  T  C        120        1 0.495833333   33
rs6039167         20   846271      +  G  A        120        1 0.012500000  117
rs6037443         20  2909408      +  G  A        120        1 0.491666667   35
rs7261762         20   853448      +  A  G        120        1 0.008333333  118
rs554362          20   850002      +  A  G        120        1 0.012500000  117
rs7273309         20   853154      +  C  T        120        1 0.004166667  119
rs7267882         20   853785      +  G  A        120        1 0.004166667  119
rs8123328         20   855045      +  G  T        120        1 0.004166667  119
rs7265788         20   855426      +  A  G        120        1 0.004166667  119
rs2325971         20  2774477      +  T  C        120        1 0.470833333   38
rs873711          20  2775468      +  A  G        120        1 0.470833333   38
rs6037425         20  2785379      +  G  C        120        1 0.470833333   38
          P.12 P.22    Pexact          Fmax      Plrt
rs566570    55   32 0.3648758  0.083269672 0.3614010
rs6039167    3    0 1.0000000 -0.012658228 0.8454919
rs6037443   52   33 0.1467972  0.133092526 0.1442654
rs7261762    2    0 1.0000000 -0.008403361 0.8968501
rs554362     3    0 1.0000000 -0.012658228 0.8454919
rs7273309    1    0 1.0000000 -0.004184100 0.9484250
rs7267882    1    0 1.0000000 -0.004184100 0.9484250
rs8123328    1    0 1.0000000 -0.004184100 0.9484250
rs7265788    1    0 1.0000000 -0.004184100 0.9484250
rs2325971   51   31 0.1032427  0.147097763 0.1065638
rs873711    51   31 0.1032427  0.147097763 0.1065638
rs6037425   51   31 0.1032427  0.147097763 0.1065638
```

```
> mlinfo[which(mlinfo$SNP %in% inBoth),]
```

```
           SNP Al1 Al2  Freq1    MAF Quality    Rsq
1009 rs6039167   G   A 0.9875 0.0125  0.9995 0.9800
1018  rs554362   A   G 0.6496 0.3504  0.5748 0.2481
1022 rs7273309   C   T 0.9985 0.0015  0.9970 0.1896
```

```
1023 rs7261762   A   G 0.9913 0.0087  0.9991 0.9472
1024 rs7267882   G   A 0.9911 0.0089  0.9986 0.9230
1025 rs8123328   G   T 0.9902 0.0098  0.9964 0.8126
1026 rs7265788   A   G 0.9886 0.0114  0.9929 0.6884
3877 rs2325971   T   C 0.5811 0.4189  0.8526 0.7807
3880  rs873711   A   G 0.5888 0.4112  0.8597 0.7934
3886 rs6037425   G   C 0.5871 0.4129  0.8651 0.8034
3958 rs6037443   A   G 0.4975 0.4975  0.8918 0.8340
4009  rs566570   C   T 0.5307 0.4693  0.8814 0.8074
```

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 favorable 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-`GenABEL-package`*mma* 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 8 *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 9 *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-GenABEL-package*mma* 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 ("Genome-wide association analysis", page 101), and split that is two parts.

If you did not do this yet, start R and load `GenABEL-package` library, which you will need it to work with GWA-data

```
> library(GenABEL)
```

Load the data with

```
> load("data2.RData")
```

and then split it in two parts:

```
> nids(data2)
```

```
[1] 124
```

```
> mdta1 <- data2[1:40,]
> mdta2 <- data2[41:nids(data2),]
```

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-package` 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(phdata(mdta1)$bmi)

        Shapiro-Wilk normality test

data:  phdata(mdta1)$bmi
W = 0.9328, p-value = 0.0199
```

with identical results from the transformed variable:

```
> shapiro.test(zbmi0)

        Shapiro-Wilk normality test

data:  zbmi0
W = 0.9328, p-value = 0.0199
```

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)
```

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

```
        Shapiro-Wilk normality test

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

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 (as, e.g., requested by consortium):

- **name: SNP name**
- **chromosome: chromosome number**
- **position: physical position of the SNP**
- **refallele: reference allele**
- **codedallele: coded (effect) allele**
- **strand: strand**
- **refallfreq: frequency of the reference allele**

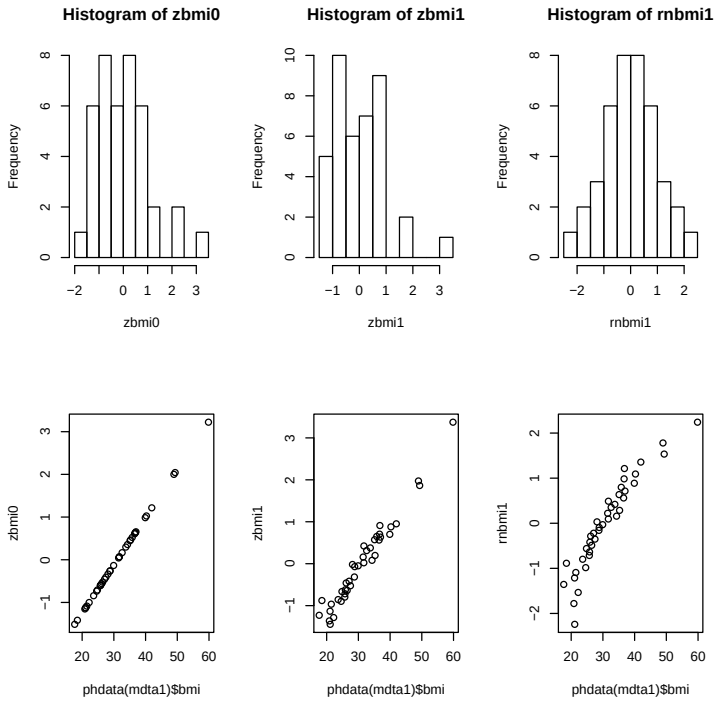

Figure 10.1: Histogram of transformed BMI and scatter-plots of the raw BMI against transformed BMI. Column 1: Z-transformation without covariates. Column 2: Z-transformation with adjustment for age and sex. Column 3: Rank-transformation to normality, after adjustment for sex and age.

- **n: number of people with data available for this SNP test**
- **beta: estimate of the effect of the allele**
- **se_beta: standard error of the effect estimate**
- **p: P-value for the test**
- **p_corr: corrected P-value (we will use Genomic Control)**
- **call: SNP call rate**
- **phwe: P-value from the exact test for HWE**

Let us look what we get as an output from `qtscore` analysis:

```
> results(qts1)[1:2,]
```

```
          Chromosome Position Strand A1 A2  N       effB   se_effB  chi2.1df
rs1646456          1      653      +  C  G 40  0.1768955 0.2560635 0.4772414
rs4435802          1     5291      +  C  A 40 -0.4064900 0.4095768 0.9849838
              P1df       effAB     effBB  chi2.2df      P2df     Pc1df
rs1646456 0.4896745  0.09703712 0.4909062 0.6323627 0.7289272 0.4896745
rs4435802 0.3209715 -0.40649004        NA 0.9849838 0.3209715 0.3209715
```

You can see that most infromation is already present in the output, though called using names which are different from these requested. However, we miss reference allele frequency, SNP call rate and P-value from the Hardy-Weinberg equilibrium test. These however can be computed using the `summary` function:

```
> # for data part 1:
> sum1 <- summary(gtdata(mdta1))
> sum1[1:2,]

          Chromosome Position Strand A1 A2 NoMeasured CallRate    Q.2 P.11 P.12
rs1646456          1      653      +  C  G         40        1 0.3375   16   21
rs4435802          1     5291      +  C  A         40        1 0.0875   33    7
          P.22    Pexact        Fmax      Plrt
rs1646456    3 0.4775382 -0.17400419 0.2616425
rs4435802    0 1.0000000 -0.09589041 0.4122629

> # ... and for data part 2:
> sum2 <- summary(gtdata(mdta2))
> sum2[1:2,]

          Chromosome Position Strand A1 A2 NoMeasured  CallRate        Q.2 P.11
rs1646456          1      653      +  C  G         83 0.9880952 0.33734940   36
rs4435802          1     5291      +  C  A         82 0.9761905 0.07926829   69
          P.12 P.22 Pexact        Fmax     Plrt
rs1646456   38    9      1 -0.02402597 0.826397
rs4435802   13    0      1 -0.08609272 0.289791
```

Note, however, that we now got frequency of the *effective* (or coded) allele, not the frequency of the reference allele! The quantity we need can be easily computed, though:

```
> # for data part 1
> refallfreq1 <- 1 - sum1[,"Q.2"]
> # ... and for data part 2
> refallfreq2 <- 1 - sum2[,"Q.2"]
```

At this moment we can arrange the required data frame:

```
> mdf1 <- data.frame(name=snpnames(qts1),chromosome=chromosome(qts1),
+         position=map(qts1),refallele=refallele(qts1),
+         codedallele=effallele(qts1),strand = strand(qts1),
+         refallelefreq = refallfreq1,n=qts1[,"N"],
+         beta=qts1[,"effB"],
+         se_beta=qts1[,"se_effB"],p=qts1[,"P1df"],
+         p_corr=qts1[,"Pc1df"],call = sum1[,"CallRate"],
+         phwe = sum1[,"Pexact"], stringsAsFactors = FALSE)
```

note the last argument – `stringsAsFactors = FALSE`. I suggest you use that by default when constructing a new data frame – unless you are sure that you can work out your way later on with strings saved as factors.

Let us inspect the first 5 raws of the resulting output:

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

```
             name chromosome position refallele codedallele strand
rs1646456 rs1646456          1      653         C           G      +
rs4435802 rs4435802          1     5291         C           A      +
rs946364   rs946364          1     8533         T           C      -
rs299251   rs299251          1    10737         A           G      +
rs2456488 rs2456488          1    11779         G           C      +
          refallelefreq  n       beta   se_beta         p    p_corr  call
rs1646456     0.6625000 40  0.1768955 0.2560635 0.4896745 0.4896745 1.000
rs4435802     0.9125000 40 -0.4064900 0.4095768 0.3209715 0.3209715 1.000
rs946364      0.7236842 38  0.2944870 0.2752734 0.2847102 0.2847102 0.950
rs299251      0.9615385 39  0.4958353 0.5989959 0.4077965 0.4077965 0.975
rs2456488     0.6625000 40  0.1145684 0.2271545 0.6140062 0.6140062 1.000
             phwe
rs1646456 0.4775382
rs4435802 1.0000000
rs946364  0.6911168
rs299251  1.0000000
rs2456488 0.7320709
```

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-package` for meta-analysis, you best use `formetascore` function, which basically performs operations similar to described, and reports results in format compatible with `MetABEL-package`.

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

```
> mdf1 <- formetascore(bmi~sex+age,mdta1,transform=rntransform, verbosity = 2 )
```

You can see that results are the same as previously:

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

```
             name chromosome position strand allele1 allele2   build
rs1646456 rs1646456          1      653      +       C       G unknown
rs4435802 rs4435802          1     5291      +       C       A unknown
rs946364   rs946364          1     8533      -       T       C unknown
rs299251   rs299251          1    10737      +       A       G unknown
rs2456488 rs2456488          1    11779      +       G       C unknown
          effallele effallelefreq  n       beta    sebeta         p       pgc
rs1646456         G    0.33750000 40  0.1768955 0.2560635 0.4896745 0.4896745
rs4435802         A    0.08750000 40 -0.4064900 0.4095768 0.3209715 0.3209715
rs946364          C    0.27631579 38  0.2944870 0.2752734 0.2847102 0.2847102
rs299251          G    0.03846154 39  0.4958353 0.5989959 0.4077965 0.4077965
rs2456488         C    0.33750000 40  0.1145684 0.2271545 0.6140062 0.6140062
          lambda.estimate lambda.se    pexhwe  call
rs1646456               1        NA 0.4775382 1.000
rs4435802               1        NA 1.0000000 1.000
rs946364                1        NA 0.6911168 0.950
rs299251                1        NA 1.0000000 0.975
rs2456488               1        NA 0.7320709 1.000
```

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, verbosity = 2 )
```

We can inspect the first five lines of the output with

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

```
              name chromosome position strand allele1 allele2   build
rs1646456 rs1646456          1      653      +       C       G unknown
rs4435802 rs4435802          1     5291      +       C       A unknown
rs946364   rs946364          1     8533      -       T       C unknown
rs299251   rs299251          1    10737      +       A       G unknown
rs2456488 rs2456488          1    11779      +       G       C unknown
          effallele effallelefreq  n        beta    sebeta          p
rs1646456         G    0.32926829 82 -0.04879397 0.1663650 0.76929696
rs4435802         A    0.08024691 81  0.37724197 0.2984226 0.20618701
rs946364          C    0.25903614 83 -0.14414880 0.1790329 0.42073156
rs299251          G    0.04216867 83 -0.69920378 0.3919648 0.07444912
rs2456488         C    0.34146341 82 -0.23105805 0.1520352 0.12856957
                 pgc lambda.estimate    lambda.se    pexhwe      call
rs1646456 0.77676571        1.070017 0.0009650323 0.8038996 0.9879518
rs4435802 0.22168453        1.070017 0.0009650323 1.0000000 0.9759036
rs946364  0.43635427        1.070017 0.0009650323 1.0000000 1.0000000
rs299251  0.08461892        1.070017 0.0009650323 1.0000000 1.0000000
rs2456488 0.14177789        1.070017 0.0009650323 0.4685397 0.9879518
```

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, verbosity = 2 )
> write.csv(mdf3,file="RData/part3.rnbmisexage.csv",row.names=F)
```

10.4 Meta-analysis with `MetABEL-package`

Now we are ready to meta-analyse GWA data coming from three studies. For this we will need to use `MetABEL-package` 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-package`, `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 = 0.9499903
Lambda Part2 = 1.098705
Corrected Lambda Part1 = 0.9499903
Corrected Lambda Part2 = 1
Lambda POOLED data = 1.023276
... 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 122     2
2 rs1006497      +       T       G         G          1  2658810 122     2
3 rs1011580      +       A       G         G          3 10048771 121     2
4 rs1011953      +       A       G         G          2  6464510 123     2
5 rs1013473      +       A       T         T          1  4487262 123     2
         beta    sebeta effallelefreq      call     pexhwe  obetaPart1
1  0.11443814 0.1850338     0.1270492 0.9919020 4.64447636  0.08819818
2 -0.07946221 0.1954998     0.1434426 0.9920082 0.16813927 -0.25425751
3 -0.06801448 0.1397522     0.5041322 0.9837773 3.15056880 -0.06755996
4  0.07539928 0.1372443     0.3292683 1.0000000 0.10373542  0.18982068
5  0.30393338 0.1334451     0.5121951 1.0000000 0.04821391  0.31522388
     obetaPart2  osePart1  osePart2      chi2         p
1  0.1230224623 0.3726827 0.2131624 0.3825070 0.5362646
2 -0.0001117686 0.3498751 0.2357343 0.1652071 0.6844070
3 -0.0683575796 0.2130800 0.1851327 0.2368565 0.6264858
4  0.0055254619 0.2228918 0.1741797 0.3018185 0.5827446
5  0.2978861467 0.2259542 0.1653644 5.1874252 0.0227509
```

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 = 1.340905
... 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 310     3
5 rs1006497      +       T       G         G          1  2658810 315     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  0.11824774 0.12085884     0.1241935 0.9780568 6.5922479  0.08819818
5 -0.02366946 0.11439872     0.1603175 0.9937465 0.2313494 -0.25425751
     obetaPart2    obetamdf3  osePart1  osePart2    osemdf3       chi2
1            NA -0.055213494        NA        NA 0.11185255 0.24366808
2            NA -0.085048708        NA        NA 0.13218866 0.41394922
3            NA -0.037120652        NA        NA 0.08523891 0.18965112
4  0.1230224623  0.121082405 0.3726827 0.2131624 0.15961076 0.95725763
5 -0.0001117686  0.005382408 0.3498751 0.2357343 0.14107317 0.04280893
          p
1 0.6215693
2 0.5199718
3 0.6632071
4 0.3278788
5 0.8360855
```

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"

Population part1 , reading RData/part1.rnbmisexage.csv done
Dimesions after filters are 3482 18
population part2, reading RData/part2.rnbmisexage.csv done
Dimesions after filters are 3530 18
analysing ...
Lambda part1 = 0.9493588
Lambda part2 = 1.095788
Corrected Lambda part1 = 0.9493588
Corrected Lambda part2 = 1
Lambda POOLED data = 1.024196
... DONE
Dimesions after pooling are 3533 20
```

```
population part3, reading RData/part3.rnbmisexage.csv done
Dimesions after filters are 7444 18
analysing ...
Lambda part3 = 1.12461
Corrected Lambda part3 = 1
Lambda POOLED data = 1.345936
... DONE
Dimesions after pooling are 7444 22
$analysed.pops
[1] "part1" "part2" "part3"
```

extra arguments regulate the SNP exclusion criteria: `maf=0.01` tells to exclude SNPs with minor allele frequency less then 0.5%, `call=0.95` tells to drop SNPs with call rate less than 95%, and `phwe=1.e-8` instructs to exclude SNPs with HWE P-value $\leq 10^{-8}$.

Now we can read and inspect the results of meta-analysis with:

```
> 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 310     3
4 rs1006497      +       T       G         G          1  2658810 315     3
5 rs1010481      +       A       C         C          2  8409087 190     1
         beta     sebeta effallelefreq      call    pexhwe  obetapart1
1 -0.08504871 0.13198426     0.8157895 0.9793814 2.8521518          NA
2 -0.03712065 0.08510711     0.5078947 0.9793814 6.8378385          NA
3  0.11825684 0.12070003     0.1241935 0.9780568 6.5922479  0.08819818
4 -0.02359567 0.11424656     0.1603175 0.9937465 0.2313494 -0.25425751
5 -0.03241475 0.12358721     0.2657895 0.9793814 0.0000000          NA
     obetapart2   obetapart3  osepart1  osepart2   osepart3       chi2
1            NA -0.085048708        NA        NA 0.13198426 0.41523235
2            NA -0.037120652        NA        NA 0.08510711 0.19023899
3  0.1230224623  0.121082405 0.3726827 0.2128792 0.15936396 0.95992605
4 -0.0001117686  0.005382408 0.3498751 0.2354211 0.14085503 0.04265582
5            NA -0.032414750        NA        NA 0.12358721 0.06879205
          p
1 0.5193256
2 0.6627178
3 0.3272055
4 0.8363747
5 0.7931037
```

10.5 Answers to the exercise

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./.67,0.93,1.08,1./.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(0.013,0.6,0.84,0.40,0.25,0.07)
```

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] 38.465034 52.216009  6.881811 20.401860 33.466060 76.607876
```

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] 0.06622101
```

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(0.013,0.6,0.84,0.40,0.25,0.07)
> 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.

10.5.1 Exercise 9:

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./.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(0.6,0.84,0.40,0.25,0.07)
```

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] 52.216009  6.881811 20.401860 33.466060 76.607876
```

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] 1.30209
```

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(0.6,0.84,0.40,0.25,0.07)
> 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.

Chapter 11

Analysis of selected region

Small data set 'srdta', which is part of the `GenABEL-package` library, will be used in this section. Start R and load the `GenABEL-package` and the data with

```
> library(GenABEL)
> data(srdta)
```

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-package

This section is outdated. By far the most used way is through the TPED

As described in section 4.1, the `GenABEL-package` gwaa.data-class consists of phenotypic data frame and an object of snp.data-class, which contains all genetic data. To import data to `GenABEL-package`, 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 the other measurements are missing (NA, Not Available).

If you need to add phenotypes to an already created `gwaa.data-class`, you can use the `add.phdata` function. 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 an "id" variable, identical to that existing in the object, and **should not** contain any other variables with names identical to those that already exist.

The second file you need should contains genotypic data. As described in section 4.1 ("General description of `gwaa.data-class`", page 77), `GenABEL-package` 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-package` provides a number of function to convert these data from different formats to the internal `GenABEL-package` 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".

First, start R and load `GenABEL-package`:

```
> library(GenABEL)
```

You can convert the data to GenABEL-package 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-package that strand information is provided in the file.

At this moment, you can load the data into GenABEL-package. Assume that the phenotypic file described above is called "phe0.dat" and the converted genotypic file in the raw GenABEL-package 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=TRUE)
```

```
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-package 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
```

. . . than check that the genotypes have imported right:

```
> g0 <- as.character(df@gtdata)
> g0
```

```
          rs1001 rs6679 rs2401 mitoA1 rs123 rs6679 rs876
cd289982  "A/A"  "G/T"  "A/A"  "A/A"  "T/T" "G/T"  NA
cd325285  "A/G"  "G/G"  "C/C"  "C/C"  "T/G" "G/G"  NA
cd357273  "A/G"  "G/G"  "C/C"  NA     "T/T" "G/G"  "T/T"
cd872422  "A/A"  "G/T"  "C/C"  NA     "T/T" "G/T"  "G/G"
cd1005389 "G/G"  "G/G"  "C/A"  NA     "T/T" "G/G"  "T/T"
```

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

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

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

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

In a real Illumina file, a coding on the TOP strand is supplied. Then, the file will normally look like

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

and the conversion command will be

```
> convert.snp.illumina(inf="RData/gen0.illuwos",
+                      out="RData/gen0iwos.raw",
+                      strand="+")
```

```
Reading genotypes from file 'RData/gen0.illuwos' ...
Writing to file 'RData/gen0iwos.raw' ...
... done.
```

In this particular data set, after conversion, the "+" strand will actually mean not "forward", but TOP – something you should remember for this particular data. The resulting file will look like this:

You can load the data with

```
> df <- load.gwaa.data(phe="RData/phe0.dat",
+                      gen="RData/gen0iwos.raw",
+                      force=TRUE)

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

> as.character(df@gtdata@strand)

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

> as.character(df@gtdata@coding)

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

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

```
> g0 == g1

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

A.2 Converting PLINK tped files

PLINK TPED (transposed-pedigree) format bears notable similarity to what we call "Illumina" format, with few exceptions: (1) there is no header line giving field names (and therefore IDs are stored in a separate file) (2) the first column gives chromosome, second – SNP name, third *genetic* map (usually kept as zeroes), the fourth – physical position, and, starting with the fifth column, genotypic data are listed, (3) finally, within a genotypes, alleles are separated with a space. In TPED format, the data we already worked with would look like

```
 1   rs1001   0    1235   A A   A G   A G   A A   G G
 9   rs6679   0    2344   G T   G G   G G   T G   G G
22   rs2401   0    3455   A A   C C   C C   C C   A C
 X    rs123   0   32535   T T   G T   T T   T T   T T
XY   rs6679   0    2344   G T   G G   G G   T G   G G
 Y    rs876   0   23556   0 0   0 0   T T   G G   T T
mt   mitoA1   0   24245   A A   C C   0 0   0 0   0 0
```

Obviously, a separate file is needed to keep correspondence between genotypes and IDs. This file emulated standard pedigree file without a header line. The file, conventionally called a TFAM-file, should contain six columns, corresponding to pedigree ID, ID, father, mother, sex, and affection. Only the second column is used by `GenABEL-package` – 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-package` 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=TRUE)
```

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

> as.character(df@gtdata@strand)

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

> as.character(df@gtdata@coding)

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

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

```
> g0 == g1

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

A.3 Converting linkage-like files

Linkage-like files, also known as pre-makeped files, or pedigree files, represent a historic format which dates back to the time when only few markers could be typed – thus the number of subjects was usually greater than the number of markers. In that situation, it was natural and obvious to keep IDs in rows and markers in columns. In the first six columns, standard linkage-like file would contain pedigree ID, ID, father's ID, mother's ID, sex (coded as 1 = male and

2 = female), and affection status (0 = unknown, 1 = unaffected, 2 = affected). In the following columns, genotypic information is provided. Alleles of the same genotype could be separated by a space, or by a slash ("/"). Thus the data we are working with could be presented as

```
1  cd289982 0 0 1 0   A A   G T   A A   T T   G T   0 0   A A
1  cd325285 0 0 1 0   A G   G G   C C   G T   G G   0 0   C C
1  cd357273 0 0 1 0   A G   G G   C C   T T   G G   T T   0 0
1  cd872422 0 0 1 0   A A   T G   C C   T T   T G   G G   0 0
1 cd1005389 0 0 1 0   G G   G G   A C   T T   G G   T T   0 0
```

As you can see, this file misses header line, and information what are the SNP names, position, etc. should be provided in a separate MAP-file. `GenABEL-package` 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-package` format with

```
> convert.snp.ped(pedfile="RData/gen0.ped",
+                 mapfile="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=TRUE)

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

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

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

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

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

```
> g0 == g1
```

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

If you are willing to import strand information, you can make use of the *extended* map format. In this format the strand information is added to the map-file:

```
chr  name    pos    strand coding
1    rs1001  1235   +      AG
9    rs6679  2344   +      TG
22   rs2401  3455   +      AC
X    rs123   32535  -      GT
XY   rs6679  2344   -      GT
Y    rs876   23556  +      GT
mt   mitoA1  24245  -      AC
```

The data can be converted to the internal `GenABEL-package` format with

```
> convert.snp.ped(pedfile="RData/gen0.ped",
+                 mapfile="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=TRUE)

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

> as.character(df@gtdata@strand)

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

> as.character(df@gtdata@coding)

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

> g0 == g1

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

A.4 Converting from MACH format

The data from MACH format can be converted by using `convert.snp.mach`. This function actually calls `convert.snp.ped` in specific format. MACH software is widely used for SNP imputations. For our needs we consider two files produced by MACH: pedigree file with (the imputed) genotypic data, and info-file, containing information about quality of imputations for particular SNP.

SEE HELP FOR `convert.snp.mach` for further details.

A.5 Converting from text format

Appendix B

GenABEL internals

B.1 Internal structure of `gwaa.data-class`

Start R and load `GenABEL-package` library using command

```
> library(GenABEL)
```

After that, load the data with the command

```
> data(srdta)
```

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-package`, special data class, `gwaa.data-class` is used to store GWA data. The structure of this data class is shown at the figure B.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 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

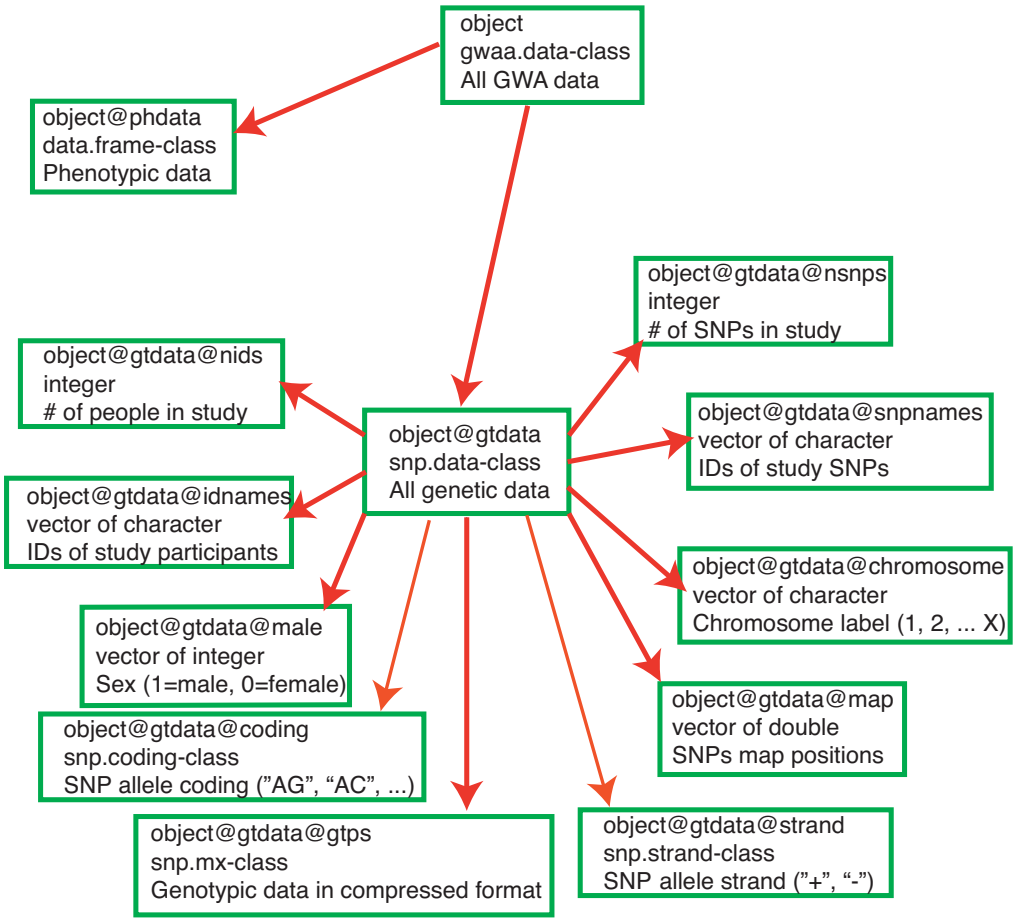

Figure B.1: Structure of gwaa.data-class. In every box, first line contains the object and slot names, second line describes the class of this object, and third line describes what information is contained.

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-package` 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 B.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-package` 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-package` 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-package` uses special data class, `gwaa.data-class`, to store GWA data.

Bibliography

Zaykin, D. V., Z. Meng, and M. G. Ehm, 2006 Contrasting linkage-disequilibrium patterns between cases and controls as a novel association-mapping method. Am J Hum Genet **78**: 737–746.