GenABEL tutorial

Yurii Aulchenko

February 12, 2007

Contents

1 Introduction

1.1 Overview

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

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

If you read this tutorial not as a part of the GE03 course, and you are eager to start with you GWA analysis without reading all the not-so-strictly-necessary staff, start directly from the section 4 ("Phenotypic and genotypic QC").

For the course participants, and the ones who decided to learn all details, be prepared for long exercise. During this exercise you will cover data structures used in `GenABEL`, and R and `GenABEL` commands, which are helpful in analysis of GWA data. To make the exercise fast and simple, we will analyse data on small region spanning about 2.5 Mb and containing 833 single-nucleotide polymorphisms (SNPs) only. However, the logic of this exercise is extendible to a whole-genome scan.

Association between several traits (one binary and several quantitative) and SNPs covering the region will be analysed during the exercise.

First, in the section 2 ("Working with gwaa.data-class") you will learn about the `gwaa.data-class`, which is used to store GWA data in GenaABEL. Next, you will cover some R basics, such as sub-setting of vectors and matrices (section 2.2, "Sub-setting vectors and matrices") and data frames (section 2.3, "Working with data frames"). The last subsection (2.4, "Sub-setting and coercing gwaa.data") of section 2 shows how one can do sub-setting of `gwaa.data-class` objects. You will need that to manage your data, e.g. select and do some region-specific analyses or perform analyses in subsets of your study population.

In the next section, 3 ("Simple analysis of phenotypic and genotypic data"), you will learn how to use R to do simple analyses (tables, testing distributions, regression...) of your data.

Section 4 ("Phenotypic and genotypic QC") will exemplify basic phenotypic and genotypic Quality Control (QC) procedures implemented in `GenABEL`.

Section 5 ("Genome-wide association analysis") is the core of the tutorial; it shows how GWA analysis is done using `GenABEL`.

The last section, 6 ("Analysis of selected region"), is dedicated to analysis of haplotype association and analysis of SNP interactions.

Answers to exercises are provided in section C.

Experienced R users may take the following sequence: start with the next subsection (1.2, "Loading `GenABEL` and example data") and read general description of the `gwaa.data-class` (subsection 2.1, "General description of `gwaa.data-class`"), then jump to the subsection 2.4 ("Sub-setting and coercing gwaa.data"). Skip section 3 ("Simple analysis of phenotypic and genotypic data"), and then continue from section 4 ("Phenotypic and genotypic QC") on.

1.2 Loading `GenABEL` and example data

Before you can start with the exercise, you need to start R and load `GenABEL` library using command

```
> library(GenABEL)
```

Please mind capital and lower case letters – otherwise the library will not load.

After that, you can load the data with the command

```
> data(srdta)
```

If you did everything right, you should be able to inspect the data space with `ls()` and see the existing data objects:

```
> ls()
```

```
[1] "srdta"
```

This output says you that one data object, `srdta`, was loaded.

Summary:

- `GenABEL` is loaded using command `library(GenABEL)`.
- You can see loaded data objects using command `ls()`.

2 Working with gwaa.data-class

2.1 General description of `gwaa.data-class`

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

In GWA analysis, different types of data are used. These include the phenotypic and genotypic data on the study participants and chromosome and location of every SNP. One could attempt to store all phenotypes and genotypes together

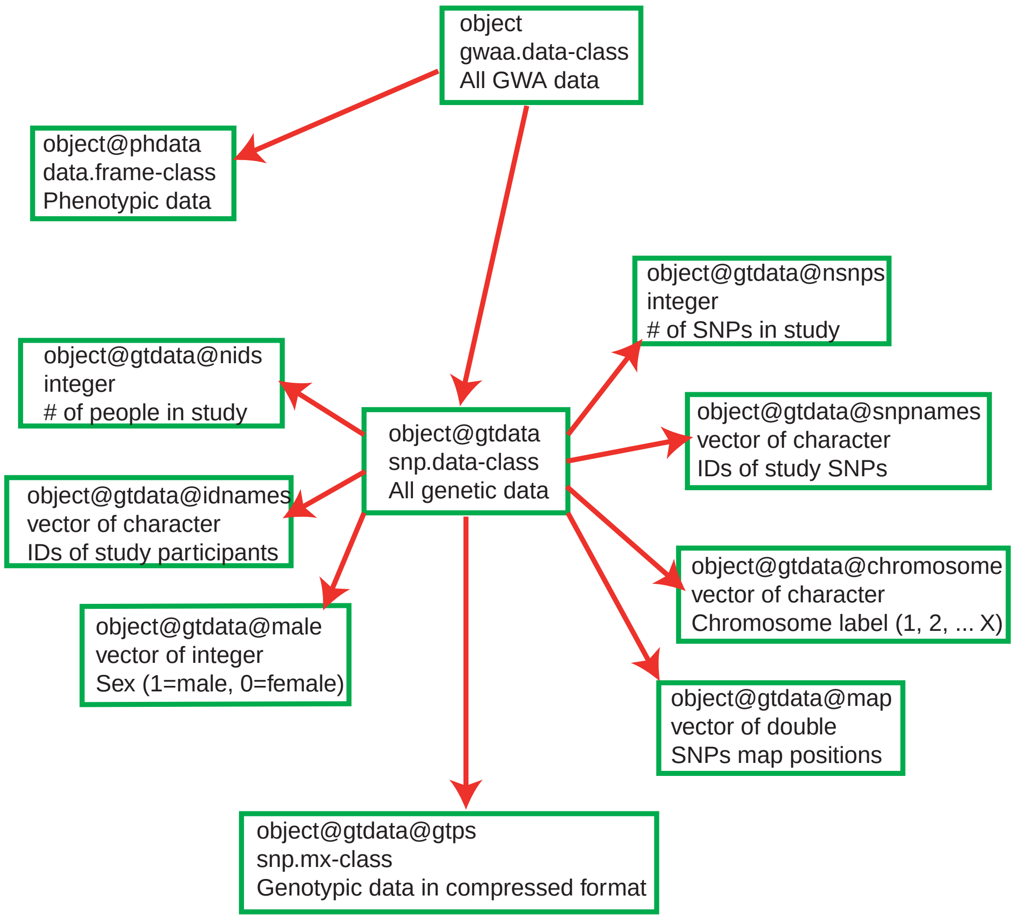

Figure 1: Structure of gwaa.data-class. In every box, first line contains the object and slot names, second line describes the class of this object, and third line describes what information is contained.

in a single table, using, e.g. one row per study subject; than the columns will correspond to study phenotypes and SNPs. For a typical GWA data set, this would lead to a table of few thousands rows and few hundreds of thousands of columns. Such a format is generated when one downloads HapMap data for a region. To store GWA data in such tables internally, within R, proves to be inefficient. In `GenABEL`, special data class, `gwaa.data-class` is used to store GWA data. The structure of this data class is shown at the figure 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), and slot `gtdata`, which contains all GWA genetic information in an object of class `snp.data-class` (figure 1). This class, in turn, has slots `nids`, containing the number of study subjects, `idnames`, containing all ID names of these subjects, `male`, containing the sex code for the subjects (1=male, 0=female), `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.

If 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.

Before you start with the exercise, you will need to learn about several useful functions. First function, `length(A)`, returns the total length (the number of elements) of vector `A`. For example, the number of elements in slot `snpnames` of the slot `gtdata` of the object `srdta` must be 833 (because there are 833 SNPs described). To get this result, you can run:

```
> length(srdta@gtdata@snpnames)
```

```
[1] 833
```

Other useful function taking the sum over all elements of a vector `A` is `sum(A)`. If there are missing values in `A`, the sum over non-missing elements may be computed with `sum(A,na.rm=TRUE)`. For example:

```
> a <- c(0, 1, 2)
> sum(a)
```

```
[1] 3
```

```
> b <- c(0, 1, 2, NA)
> sum(b)
```

```
[1] NA
```

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

```
[1] 3
```

Function `mean(A)` returns the mean of the values of `A`. If there are missing values in `A`, the mean over non-missing members may be computed with `mean(A,na.rm=TRUE)`. For example:

```
> mean(a)
```

```
[1] 1
```

```
> mean(b)
```

```
[1] NA
```

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

[1] 1
```

Meanwhile, we have created new data entries `a` and `b`. At any time, you can see all data objects by using `ls()`:

```
> ls()

[1] "a"     "b"     "srdta"
```

A help page on any of the R functions may be accessed via `help(function)`. For example, to get help on the `sum()` function, type `help(sum)`. Please check the help on R functions `sum()` and `mean()`. You can also read general man page on `GenABEL` using `help(GenABEL)`. To see help on `gwaa.data-class`, you can use `help("gwaa.data-class")` (mind the quotation marks!).

Exercise 1 *Explore the content of different slots of* `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?*

Summary:

- An object of some class has "slots" which may contain actual data or objects of other classes. The information stored at a particular `slot` of an `object` can be accessed by command `object@slot`.
- `GenABEL` uses special data class, `gwaa.data-class`, to store GWA data.
- Function `length(A)` returns number of elements in `A`.
- Function `sum(A)` returns sum of elements in `A`. Function `sum(A,na.rm=TRUE)` returns sum of measured elements of `A`.
- Function `mean(A)` returns mean of `A`. Function `mean(A,na.rm=TRUE)` returns mean of `A` over the measured elements.

2.2 Sub-setting vectors and matrices

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

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

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

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

```
> a[3]
```

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

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

```
> a[2:4]
```

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

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

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

```
[1] 1 3 5
```

```
> a[dje]
```

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

Exercise 2 *Explore the* `gtdata` *slot of the* `srdta`

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

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

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

[1] 150 175 182 173 192 168

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

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

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

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

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

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

```
> tmphgt[vec]

[1] 175 182 173 192
```

or you can get IDs of these people with

```
> tmpids[vec]

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

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

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

[1] FALSE  TRUE  TRUE  TRUE FALSE FALSE

> tmpids[vec]

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

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

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

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

[1] 2 3 4

> tmpids[vec]

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

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

```
> ls()

[1] "a"        "b"        "dje"      "srdta"    "tmphgt" "tmpids" "vec"

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

and check if the removal was successful:

```
> ls()

[1] "a"      "b"      "dje"    "srdta" "vec"
```

Exercise 3 *Explore slot containing map (`map`) and slot containing SNP names (`snpnames`) of the `gtdata` slot of `srdta`.*

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

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

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

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

[2]Because it treats NAs for you

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

```
> a[2, 2]
```

```
[1] 22
```

...the second row and third column:

```
> a[2, 3]
```

```
[1] 32
```

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

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

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

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

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

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

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

```
> a[1, ]
```

```
[1] 11 21 31
```

...and third column

```
> a[, 3]
```

```
[1] 31 32 33
```

...or columns 1 and 3:

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

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

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

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

```
> a > 21

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

or obtain these elements by

```
> a[a > 21]

[1] 22 23 31 32 33
```

Summary:

- It is possible to get sub-sets of vectors and matrices by specifying index value or a logical condition (of the same length as the vector / matrix) between square brackets (`[, ]`)
- When you obtain an element of a matrix with `[i,j]`, `i` is the row and `j` is the column of the matrix.
- Function `which(A)` returns index of the elements of A which are "true".

2.3 Working with data frames

The phenotypic data are stored in the `phdata` slot, which is a standard R data frame. This could be thought of as a matrix, though it has some extra attractive features. In phenotypic data frame `phdata`, rows correspond to subjects and columns correspond to variables.

Exercise 4 *Explore* `phdata` *slot of* `srdta`*.*

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

The nice feature of data frame is that columns carry names for the variables, and the data stored there can be retrieved by referencing these names. To see what are the variable names, we can run

```
> names(srdta@phdata)
```

```
[1] "id"  "sex" "age" "qt1" "qt2" "qt3" "bt"
```

These variables correspond to personal ID, sex, age, and three quantitative and one binary traits. A variable from a data frame `frame`, which has some name `name` can be accessed through `frame$name`. This will return a conventional vector, which may be further sub-setted.

Let us check what are the ID and sex for person 75:

```
> srdta@phdata$id[75]
```

```
[1] "p75"
```

```
> srdta@phdata$sex[75]
```

```
[1] 0
```

We can also easily check how many people are described in the data set

```
> length(srdta@phdata$sex)
```

```
[1] 2500
```

how many of these are males

```
> sum(srdta@phdata$sex)
```

```
[1] 1275
```

and what is male sex proportion:

```
> sum(srdta@phdata$sex)/length(srdta@phdata$sex)
```

```
[1] 0.51
```

You can avoid typing long names by "attaching" the data frame to the search path:

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

After that, you can access the variables in the data set directly, e.g.

```
> sum(sex)
```

```
[1] 1275
```

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

```
[1] 0.51
```

If you modify a variable in a data frame named say `frame`, for the changes to take effect, you will need to detach the data frame by `detach(frame)` and then re-attach it by `attach(frame)`

As well as with vectors, it is possible to sub-set elements of a data frame based on (a combination of) logical conditions. For example, if we want to check who are the people with `qt1` over 2.7, we can run

```
> vec <- which(qt1 > 2.7)
> vec

[1]  44 376 688

> id[vec]

[1] "p44"  "p376" "p688"
```

Exercise 5 *Explore* `phdata` *slot of* `srdta`

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

Summary:

- A variable with name `name` from a data frame `frame`, can be accessed through `frame$name`.
- You can attach the data frame to the search path by `attach(frame)`. Then the variables contained in this data frame may be accessed directly.

2.4 Sub-setting and coercing gwaa.data

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

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

  id sex  age   qt1    qt2  qt3 bt
1 p1   1 43.4 -0.58   4.46 1.43  0
2 p2   1 48.2  0.80   6.32 3.90  1
3 p3   0 37.9 -0.52   3.26 5.05  1
4 p4   1 53.8 -1.55 888.00 3.76  1
5 p5   1 47.5  0.25   5.70 2.89  1
@nids = 5
```

```
@nsnps = 3
@nbytes = 2
@idnames = p1 p2 p3 p4 p5
@snpnames = rs10 rs18 rs29
@chromosome = 1 1 1
@map = 2500 3500 5750
@male = 1 1 0 1 1
@gtps =
     [,1] [,2] [,3]
[1,]   55   55   40
[2,]   40   80   40
```

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

However, the slot `gtps`, which actually contain the SNP data,is not readable, because the information is compressed. To get human-readable genotypes, and object of class `snp.data-class` (e.g. `srdta@gtdata`) should be coerced to a more readable format. For example, function `as.character()` will convert the data to character representation:

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

   rs10  rs18  rs29
p1 "A/A" "A/A" "A/A"
p2 "A/A" "A/A" ""
p3 "A/A" "A/A" ""
p4 "A/A" "A/A" ""
p5 "A/A" "A/B" "A/A"
```

Other useful coercion is to ”numeric”:

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

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

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

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

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

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

  rs10 rs18 rs29
1  A/A  A/A  A/A
2  A/A  A/A <NA>
3  A/A  A/A <NA>
4  A/A  A/A <NA>
5  A/A  A/B  A/A
```

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

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

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

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

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

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

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

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

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

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

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

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

```
@nbytes = 1
@idnames = p141 p147 p2000
@snpnames = rs10 rs29
@chromosome = 1 1
@map = 2500 5750
@male = 0 0 0
@gtps =
     [,1] [,2]
[1,]   58   58
```

The problem is that you get genotypes in a compressed format. To see the actual genotypes, use

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

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

or

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

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

Exercise 6 *Explore genotypes for SNP "rs114".*

1. *What is the frequency of B allele (coded as "1") in total sample?*
2. *What is the frequency of B allele in male?*
3. *What is the frequency of B allele in female?*

Summary:

- It is possible to obtain subsets of objects of `gwaa.data-class` and `snp.data-class` using standard 2D sub-setting model `[i,j]`, where `i` corresponds to study subjects and `j` corresponds to SNPs.
- It is possible to provide ID and SNP names instead of indexes when sub-setting an object of class `gwaa.data-class`.

- Function `as.numeric()` converts genotypic data from `snp.data-class` to regular integer numbers, which can be used in analysis with R.
- Function `as.character()` converts genotypic data from `snp.data-class` to character format.
- Function `as.genotype()` converts genotypic data from `snp.data-class` to the format used by library `genetics`.
- Function `as.hsgeno()` converts genotypic data from `snp.data-class` to the format used by library `haplo.stats`.

3 Simple analysis of phenotypic and genotypic data

In section ?? ("??") you have already learned about such useful R functions and `length()`, `sum()` and `mean()`. In this section, we will first learn about other simple functions which are useful for analysis of binary and quantitative traits, and about regression analysis in R.

3.1 Exploring banary traits

3.2 Exploring quantitative traits

The function `summary()` generates a summary statistics for an object. For example, to see summary for trait `qt1`, we can use

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

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

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

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

(resulting histogram is shown at figure 2)

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

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

(resulting plot is shown at figure 3)

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

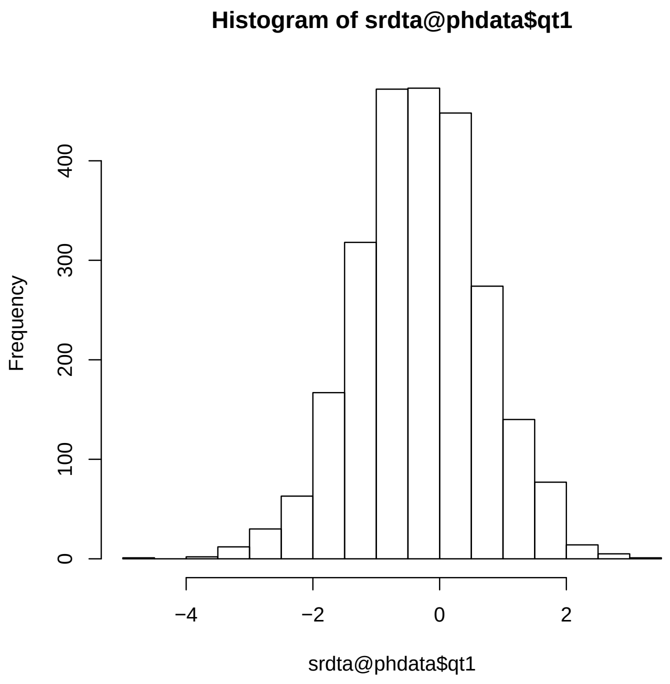

Figure 2: Histogram of qt1

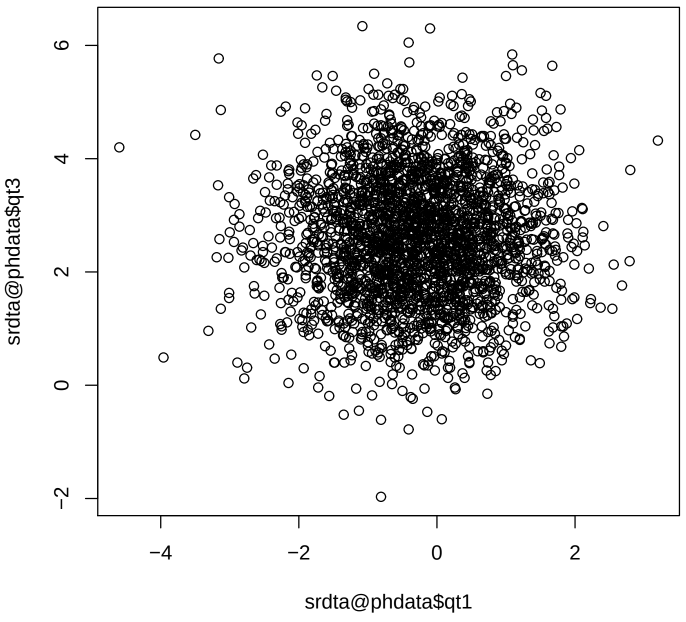

Figure 3: Scatter-plot of qt1 against qt3

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

3.3 Regression analysis

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

```
> summary(glm(srdta@phdata$qt2 ~ srdta@phdata$age + srdta@phdata$sex))

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

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

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

(Dispersion parameter for gaussian family taken to be 935.722)

    Null deviance: 2340050  on 2499  degrees of freedom
Residual deviance: 2336498  on 2497  degrees of freedom
AIC: 24203

Number of Fisher Scoring iterations: 2
```

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

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

```
> attach(srdta@phdata)

        The following object(s) are masked from srdta@phdata ( position 3 ) :

         age bt id qt1 qt2 qt3 sex
```

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

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

```
Call:
glm(formula = qt2 ~ age + sex)

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

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

(Dispersion parameter for gaussian family taken to be 935.722)

    Null deviance: 2340050  on 2499  degrees of freedom
Residual deviance: 2336498  on 2497  degrees of freedom
AIC: 24203

Number of Fisher Scoring iterations: 2
```

with the same results.

To figure out if your binary trait (`bt`) is associated with sex and age, you need to tell that this is binary trait:

```
> summary(glm(bt ~ age + sex, family = binomial()))
```

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

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

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 4
```

There is strong association between `bt` and sex and age.

4 Phenotypic and genotypic QC

4.1 Exploring genetic data

Function `summary()` is the one most useful in genetic data exploration and QC. This function generates some useful summaries for `gwaa.data-class`:

```
> summary(ssubs)

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

$gtdata
     NoMeasured CallRate Q.2 P.11 P.12 P.22 Pexact Chromosome
rs10          5      1.0 0.0    5    0    0      1          1
rs18          5      1.0 0.1    4    1    0      1          1
rs29          2      0.4 0.0    2    0    0      1          1
```

In the first section, the summary is generated for phenotypic data. In the second section, summary is generated for genotypic data. `Pexact` refers to exact P-value for the test of Hardy-Weinberg equilibrium.

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

```
> vec <- which(age >= 65)
> vec

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

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

     NoMeasured CallRate       Q.2 P.11 P.12 P.22    Pexact Chromosome
rs10         48     0.96 0.1354167   36   11    1 1.0000000          1
rs18         47     0.94 0.2765957   25   18    4 0.7245853          1
rs29         45     0.90 0.1555556   32   12    1 1.0000000          1
```

a vector of logical values:

```
> vec <- (age >= 65)
> summary(srdta@gtdata[vec, 1:3])
```

```
     NoMeasured CallRate       Q.2 P.11 P.12 P.22    Pexact Chromosome
rs10         48     0.96 0.1354167   36   11    1 1.0000000          1
rs18         47     0.94 0.2765957   25   18    4 0.7245853          1
rs29         45     0.90 0.1555556   32   12    1 1.0000000          1
```

a list with IDs of study subjects:

```
> vec <- (age >= 65)
> vec1 <- srdta@gtdata@idnames[vec]
> vec1
```

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

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

```
     NoMeasured CallRate       Q.2 P.11 P.12 P.22    Pexact Chromosome
rs10         48     0.96 0.1354167   36   11    1 1.0000000          1
rs18         47     0.94 0.2765957   25   18    4 0.7245853          1
rs29         45     0.90 0.1555556   32   12    1 1.0000000          1
```

Exercise 8 *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)*

Summary:

- When `summary()` function is applied to an object of `snp.data-class`, it return summary statistics, including exact test for Hardy-Weinberg equilibrium.
- Function `summary(A)` generates some summary statistics for variable `A`.
- Function `sd(A)` returns standard deviation of variable `A`.

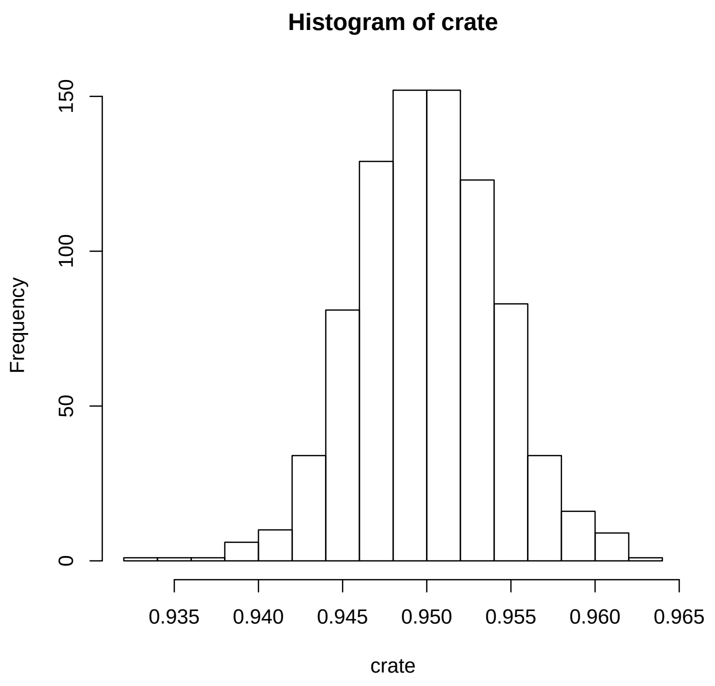

Figure 4: Histogram of the call rate

4.2 Genotypic QC

Several pieces of information are critical in accessing quality of genetic data. These include SNP call rates (per SNP and per person), distribution of minor allele frequency (MAF) and significance of deviation from Hardy-Weinberg equilibrium (HWE).

Based on `summary()` function described in previous subsection, you can generate distribution of these important characteristics.

Let us analyse the distribution of call rate first. For this, you need to get a vector of call rates:

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

At this point, you can produce a histogram of call rates by

```
> hist(crate)
```

The resulting histogram is shown at the figure 4.

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

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

```
     X<=0.93 0.93<X<=0.95 0.95<X<=0.99 X>0.99
No         0  415.0000000  418.0000000      0
Prop       0    0.4981993    0.5018007      0
```

Similar procedure may be applied to see deviation from HWE:

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

```
     X<=6.00240096038415e-05 6.00240096038415e-05<X<=0.01 0.01<X<=0.05
No               2.000000000                  7.000000000  23.00000000
Prop             0.002400960                  0.008403361   0.02761104
     0.05<X<=0.1       X>0.1
No   31.00000000 770.0000000
Prop  0.03721489   0.9243697
```

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

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

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

```
     X<=6.00240096038415e-05    X<=0.01      X<=0.05       X<=0.1 X>0.1
No               2.000000000 9.00000000 32.00000000 63.00000000   833
Prop             0.002400960 0.01080432  0.03841537  0.07563025     1
```

The same logic applies for the MAF distribution. First, derive MAF:

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

Next, you can generate histograms for frequency and MAF:

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

(shown at the figure 5) and then generate table describing frequency distribution:

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

```
         X<=0.01 0.01<X<=0.05 0.05<X<=0.1  0.1<X<=0.2  0.2<X<=0.5  0.5<X<=0.8
No   22.00000000  53.00000000  99.0000000 132.0000000 313.0000000 187.0000000
Prop  0.02641056   0.06362545   0.1188475   0.1584634   0.3757503   0.2244898
      0.8<X<=0.9 0.9<X<=0.95 0.95<X<=0.99 X>0.99
No   18.00000000 8.000000000  1.000000000      0
Prop  0.02160864 0.009603842  0.001200480      0
```

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

```
     X<=0     X<=0.01    X<=0.05       X<=0.1       X<=0.2 X>0.2
No       0 22.00000000 76.0000000 183.0000000 333.0000000   833
Prop     0  0.02641056  0.0912365   0.2196879   0.3997599     1
```

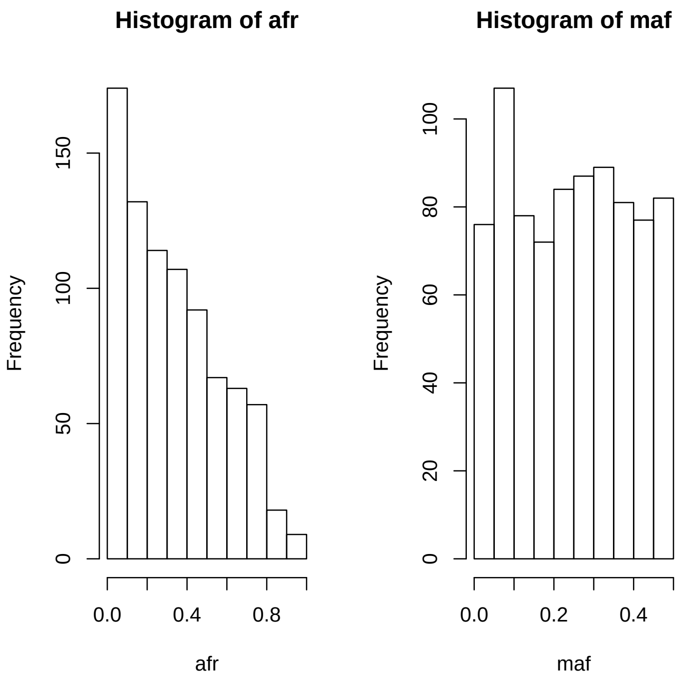

Figure 5: Histogram of the call rate

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

The outliers who have increased average heterozygosity may be suggestive of contaminated DNA samples. For the sake of time, we will demonstrate this procedure using only 100 people from the data set:

```
> het <- heteroz(srdta, ids = c(1:100))

[1] 10
[1] 20
[1] 30
[1] 40
[1] 50
[1] 60
[1] 70
[1] 80
[1] 90
[1] 100

> mean(het$mean)

[1] 0.3262500

> catable(het$mean, 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     1.00        3.00       18.00       47.00       31.00     0
Prop   0.01        0.03        0.18        0.47        0.31     0
```

Other function for genotypic QC is `check.marker()`.

```
> gQC <- check.marker(srdta, call = 0.95, maf = 0.01, fdrate = 0.05)

833 markers in total
0 (0%) markers excluded as redundant
22 (2.641056%) markers excluded as having low maf
387 (46.45858%) markers excluded because of low call rate
2 (0.2400960%) markers excluded because they are out of HWE
In total, 437 (52.46098%) markers passed all criteria

> summary(gQC)

          NoCall NoMAF NoHWE Redundant
NoCall       372    15     0         0
NoMAF         NA     7     0         0
NoHWE         NA    NA     2         0
Redundant     NA    NA    NA         0

> HWE.show(srdta, snps = gQC$nohwe)

HWE summary for rs73 :
                  A/A      A/B         B/B
observed 2331.0000000 44.00000  10.0000000
```

```
expected 2321.4293501 63.14130   0.4293501
chi2+       0.0394573  5.80269 213.3395064
Chi2 = 219.1817 ; P = 0; exact P = 1.79247e-12

HWE summary for rs128 :
                  A/A        A/B       B/B
observed 2.281000e+03 101.000000  9.000000
expected 2.273481e+03 116.038687  1.480657
chi2+    2.486959e-02   1.949023 38.186115
Chi2 = 40.16001 ; P = 2.339896e-10; exact P = 9.4086e-06
```

Or the same with HWE check only in controls:

```
> gQC <- check.marker(srdta, hweids = (srdta@phdata$bt == 0), call = 0.95,
+     maf = 0.01, fdrate = 0.05)
```

```
833 markers in total
0 (0%) markers excluded as redundant
22 (2.641056%) markers excluded as having low maf
387 (46.45858%) markers excluded because of low call rate
2 (0.2400960%) markers excluded because they are out of HWE
In total, 437 (52.46098%) markers passed all criteria
```

```
> summary(gQC)
```

```
          NoCall NoMAF NoHWE Redundant
NoCall       372    15     0         0
NoMAF         NA     7     0         0
NoHWE         NA    NA     2         0
Redundant     NA    NA    NA         0
```

```
> HWE.show(srdta, snps = gQC$nohwe)
```

```
HWE summary for rs73 :
                  A/A      A/B         B/B
observed 2331.0000000 44.00000  10.0000000
expected 2321.4293501 63.14130   0.4293501
chi2+       0.0394573  5.80269 213.3395064
Chi2 = 219.1817 ; P = 0; exact P = 1.79247e-12

HWE summary for rs128 :
                  A/A        A/B       B/B
observed 2.281000e+03 101.000000  9.000000
expected 2.273481e+03 116.038687  1.480657
chi2+    2.486959e-02   1.949023 38.186115
Chi2 = 40.16001 ; P = 2.339896e-10; exact P = 9.4086e-06
```

Recommended minimal minor allele frequency: no power if results rely on 1 or 2 observations.

```
> gQC <- check.marker(srdta, hweids = (srdta@phdata$bt == 0), call = 0.95,
+     maf = 3/srdta@gtdata@nids, fdrate = 0.05)
```

```
833 markers in total
0 (0%) markers excluded as redundant
2 (0.2400960%) markers excluded as having low maf
387 (46.45858%) markers excluded because of low call rate
2 (0.2400960%) markers excluded because they are out of HWE
In total, 443 (53.18127%) markers passed all criteria
```

```
> summary(gQC)
```

```
          NoCall NoMAF NoHWE Redundant
NoCall       386     1     0         0
NoMAF         NA     1     0         0
NoHWE         NA    NA     2         0
Redundant     NA    NA    NA         0
```

Note: We do however suggest that only monomorphic SNPs are dropped from future analysis. All polymorphic SNPs are recommended to be taken for further analysis despite of call rate, HWE test, etc. Instead of dropping SNPs/study participants based on statistics, we would rather suggest this data is a guidance for the Laboratory.

4.3 Phenotypic QC

Phenotypic QC is an important part of an association study. You should look if the trait distribution look "reasonable"

5 Genome-wide association analysis

5.1 Analysis of binary traits

To analyse

```
> res.ccf <- ccfast("bt", data = srdta)
```

To see minimal P-values

```
> min(res.ccf$P1df)
```

```
[1] 0.009148063
```

```
> min(res.ccf$P2df)
```

```
[1] 0.004483772
```

To plot

```
> plot(res.ccf)
```

(shown at figure 6).

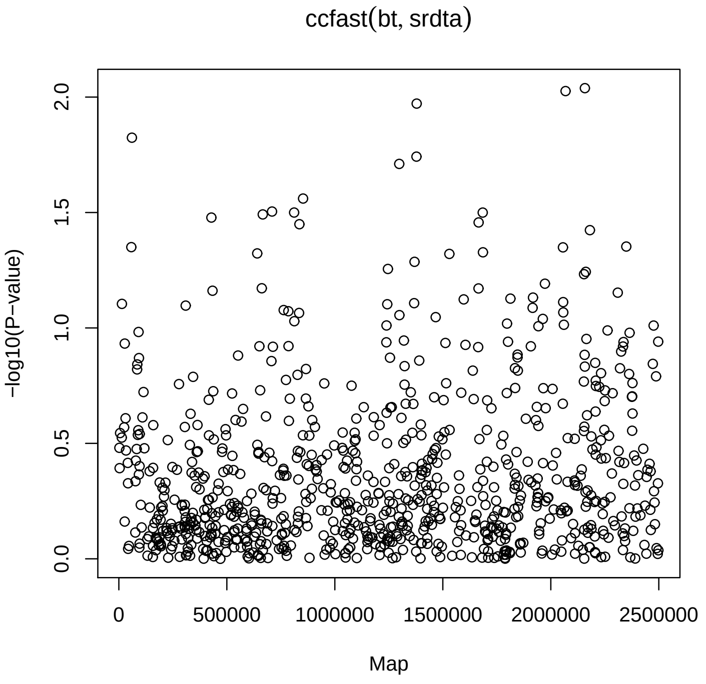

Figure 6: GWA analysis of the trait "bt"

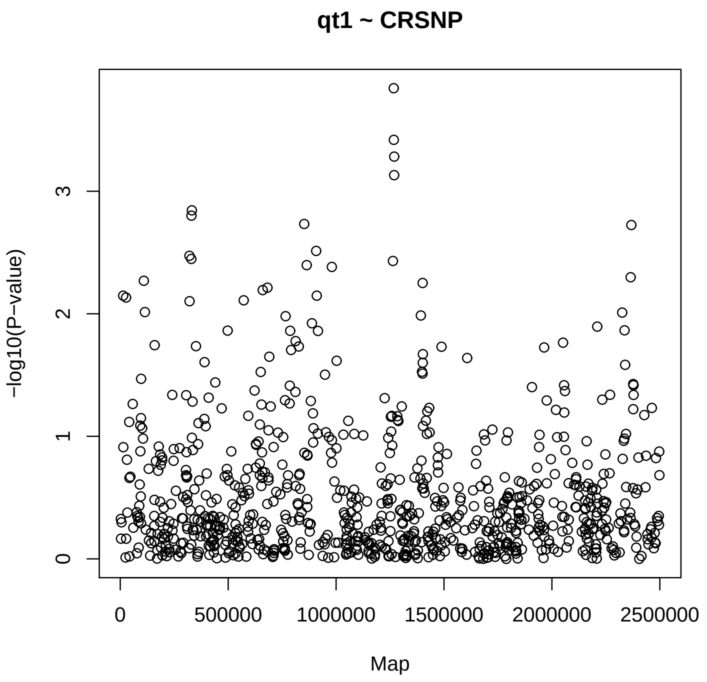

Figure 7: GWA analysis of the trait ”bt”

5.2 Analysis of quantitative traits

To analyse

```
> res.qts <- qtscore("qt1 ~ CRSNP", data = srdta)
```

To see minimal P-values

```
> min(res.qts$P1df)

[1] 0.0001442140

> min(res.qts$P2df)

[1] 0.0003874045
```

To plot

```
> plot(res.qts)
```

(shown at figure 7).

6 Analysis of selected region

6.1 Exploring linkage disequilibrium

6.2 Haplotype analysis

6.3 Analysis of interactions

6.4 Exploring public databses

A GWA analysis protocol

A.1 Genetic data descriptives and QC

Note: if you working with a data set which is already worked with, the first step must have been already done. Ask for results.
Step 1. Describe the following distributions:

- Call rate per SNP. First, generate a histogram; second, arrange a summary table. Decide on cut-off points for the table based on histogram. Suggested cut-offs (meaningful for Affymetrix) are 0.99, 0.95, 0.93, 0.9.
- Call rate per person. First, generate a histogram; second, arrange a summary table. Decide on cut-off points for the table based on histogram.
- Minor allele frequency (MAF). Generate histogram and table. MAF cut-off points: (monomorphic, MAF ≤ 0.01, $0.01 <$ MAF ≤ 0.05, $0.05 <$ MAF ≤ 0.1, $0.1 <$ MAF ≤ 0.2, MAF > 0.2)
- Exact P-value for the test for HWE. Generate QQ-plot and a table. Cut-off points: ($P{\leq}0.1$,$P{\leq}0.05$,$P{\leq}0.01$,$P{\leq}$ 0.05/(number of SNPs in study)).
- If you work with a case/control sample, repeat previous analysis for cases and controls separately.
- Generate average within-person heterozygosity by `heteroz()`; plot the histogram, make a table and check for outliers (at FDR=0.01). These outliers may indicate contaminated DNA samples.
- (not impelemented in GenABEL yet!) Generate genome-wide IBS matrix, and check for outliers. These may indicate people who belong to a different genetic population.

Step 2

- Generate a working subset by excluding monomorphic markers and people with very low call rate
- Pass the list of people with low call rate back to the Lab.

A.2 GWA analysis

Step 1
For binary traits use

- Do case-control GWA analysis using `ccfast`
- Obtain empirical GW significance using `emp.ccfast`

For QTs use

- Do QT GWA analysis using `qtscore`
- Obtain empirical GW significance using `emp.qtscore`

Step 2

- Allow for covariates in your analysis (use `qtscore` and `emp.qtscore`)

Step 3

- For each trait and analysis steps 1 and 2, generate plots showing -log10(P-value), both nominal and empirical. Show tests with 1 d.f. and 2 d.f. on the same plot using different symbols
- Select SNPs of interest. Total number of SNPs selected will depend on your results (definitely, all SNPs hitting 5% GW significance should be on the list, but you may select based on candidate genes or regions, so on.). In any case, think of at least 10 SNPs.
- Select regions of interest (the ones surrounding SNPs of interest). Depending on your population, the regional width should be from 100 Kbp to 1 Mbp. For ERGO take 200 Kbp. For GRIP take 500 Kbp. For ERF take 1 Mbp.

A.3 Regional analyses

Give summary characteristics for each SNP of interest

- Call rate
- Total genotypic distribution
- P-value for the test of HWE (total, and for cases and control separately in case-control design)
- For binary traits: genotypic distributions between cases and controls, crude and adjusted ORs. Also report 95% CI and P-values.
- For quantitative traits: crude trait means per genotype and adjusted means (or effects). Also report s.e.m. and P-values.

For each region of interest:

- Perform 2- and 3-SNP sliding window haplotype analysis using `scan.haplo`. Do not forget to adjust for covariates if you need to!
- For best slides, generate output with `haplo.score`. Use `simulate=TRUE` to obtain empirical significance.
- Generate a plot showing -log10(P-value) for the allelic and genotypic association tests, HWE test -log10(1-MAF) and haplotypic analysis results.
- Analyse association between the trait and haplotypes formed by all pairs of SNPs in the region, using `scan.haplo.2D`
- For best slides, generate output with `haplo.score`. Use `simulate=TRUE` to obtain empirical significance.
- Analyse LD in the region using function `LD`
- Plot heatmap with results from scan.haplo.2D above and LD below the diagonal.

- Analyse association between the trait and all pairs of SNPs, allowing for interaction (`scan.glm.2D`).
- Select "best" pairs based on allelic, genotypic, and interaction test P-values. For these, report P-values from allelic, genotypic, and interaction tests. Generate joint genotypic distribution (per case/control), show crude and adjusted OR (P, CI). or means per joint genotype (crude, adjusted, P, CI).
- Plot heatmap with results from scan.glm.2D (total significance above and significance of interaction below the diagonal). Do it for both allelic and genotypic tests.

B Importing data to GenABEL

C Answers to exercises

C.1 Exercise 1:

Number of people:

```
> srdta@gtdata@nids
```

```
[1] 2500
```

Number of males:

```
> sum(srdta@gtdata@male)
```

```
[1] 1275
```

Number of females:

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

```
[1] 1225
```

... or you could get the same answer like this[3]:

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

```
[1] 1225
```

The proportion of males can be computed using above results

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

```
[1] 0.51
```

or by using `mean()` function:

```
> mean(srdta@gtdata@male)
```

```
[1] 0.51
```

C.2 Exercise 2:

For the first person id is ”p1” and sex code is 1 (1=male, 2=female)

```
> srdta@gtdata@idnames[1]
```

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

```
> srdta@gtdata@male[1]
```

```
p1
 1
```

For the 22nd person id is ”p22” and sex code is 1:

[3]This is something covered later in the section 2.2 (”Sub-setting vectors and matrices”)

```
> srdta@gtdata@idnames[22]
```

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

```
> srdta@gtdata@male[22]
```

```
p22
  1
```

Among first 100 subjects, there are 53 males:

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

```
[1] 53
```

Among 4th hundred subjects there are 45 females:

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

```
[1] 45
```

Male proportion among first 1000 people is

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

```
[1] 0.508
```

Female proportion among second 1000 people is

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

```
[1] 0.476
```

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

```
> srdta@gtdata@snpnames[33]
```

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

```
> srdta@gtdata@chromosome[33]
```

```
rs422
  "1"
```

```
> srdta@gtdata@map[33]
```

```
 rs422
105500
```

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

```
> pos25 <- srdta@gtdata@map[25]
> pos25
```

```
rs365
91250
```

```
> pos26 <- srdta@gtdata@map[26]
> pos26
```

```
rs372
92750
```

```
> pos26 - pos25
```

```
rs372
 1500
```

C.3 Exercise 3:

The names of markers located after 2,490,000 b.p. are

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

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

Between 1,100,000 and 1,105,000 b.p.:

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

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

C.4 Exercise 4:

Value of the 4th variable of person 75:

```
> srdta@phdata[75, 4]
```

```
[1] -0.04
```

Value for the variable 1 is

```
> srdta@phdata[75, 1]
```

```
[1] "p75"
```

Also, if we check first 10 elements we see

```
> srdta@phdata[1:10, 1]
```

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

This is personal ID.

The sum for variable 2 is

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

```
[1] 1275
```

This is sex variable.

C.5 Exercise 5:

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

```
> sum(age > 65)
```

```
[1] 48
```

```
> vec <- which(age > 65)
> length(vec)
```

```
[1] 48
```

To get sex of these people use any of:

```
> sx65 <- sex[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 <- sex[vec]
> sx65

 [1] 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0
[39] 1 0 1 0 0 0 0 1 1 1
```

Thus, number of males is:

```
> sum(sx65)

[1] 26
```

To conclude, the proportion of male is 0.541666666666667

Distribution of qt3 in people younger and older than 65 are:

```
> summary(qt3[vec])

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

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

[1] 1.128701

> young <- which(age < 65)
> summary(qt3[young])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  -1.97    1.83    2.58    2.59    3.35    6.34   11.00 

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

[1] 1.093374
```

C.6 Exercise 6:

To compute frequency of allele B of SNP ”rs114” in total sample, you can go two ways. First, we can try to take a sum of all rs114 genotypes and divide it by twice the number of people:

```
> a <- as.numeric(srdta@gtdata[, "rs114"])
> sum(a)

[1] NA
```

This, however, returns NA, because some of the genotypes are missing. We can deal with this problem by running `sum()` with the option `na.rm=TRUE`:

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

```
[1] 559
```

However, now we do not know what was the number of people for whom the genotype was measured!

An easier way would be to compute mean value of rs114 with the `mean( ... ,na.rm=TRUE)` function and divide it by 2:

```
> mean(a, na.rm = T)/2
```

```
[1] 0.116799
```

To compute frequency of rs114 allele B in males, you can use

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

```
[1] 0.1164216
```

To compute frequency of rs114 allele B in females, you can use

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

```
[1] 0.1171942
```

Actually, the problem that we do not know how many people are measured, can be easily dealt with. This can be done by using `is.na(A)` function which returns true when some element of `A` is not measured. Thus, the number of people with measured genotype for rs114 is

```
> a <- as.numeric(srdta@gtdata[, "rs114"])
> sum(!is.na(a))
```

```
[1] 2393
```

And the allele frequency estimate is

```
> sum(a, na.rm = T)/(2 * sum(!is.na(a)))
```

```
[1] 0.116799
```

exactly the same as above.

C.7 Exercise 7:

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

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  24.10   45.10   50.00   50.04   54.80   71.60 
```

The histogram for qt2 looks strange: it seems there are few very strong outliers (figure 8) You can also see that with `summary`:

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

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

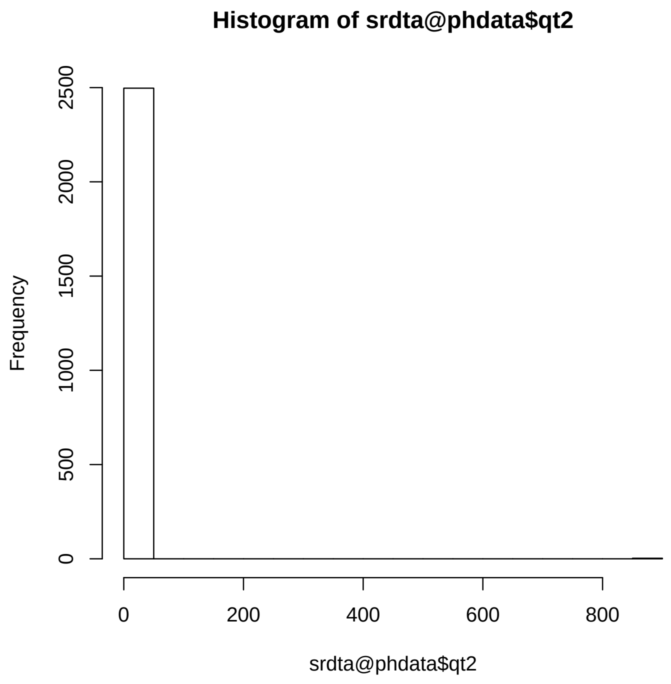

Figure 8: Histogram of qt2

C.8 Exercise 8:

To test for HWE in first 10 SNPs in total sample

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

```
      NoMeasured CallRate        Q.2 P.11 P.12 P.22       Pexact Chromosome
rs10        2384   0.9536 0.13255034 1792  552   40 7.897327e-01          1
rs18        2385   0.9540 0.28029350 1232  969  184 7.608230e-01          1
rs29        2374   0.9496 0.13774221 1763  568   43 7.955141e-01          1
rs65        2378   0.9512 0.71972246  182  969 1227 6.475412e-01          1
rs73        2385   0.9540 0.01341719 2331   44   10 1.792470e-12          1
rs114       2393   0.9572 0.11679900 1868  491   34 7.663683e-01          1
rs128       2391   0.9564 0.02488499 2281  101    9 9.408599e-06          1
rs130       2379   0.9516 0.69377890  222 1013 1144 9.615127e-01          1
rs143       2377   0.9508 0.47728229  655 1175  547 6.512540e-01          1
rs150       2369   0.9476 0.65998312  267 1077 1025 5.518478e-01          1
```

To test it in cases

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

```
      NoMeasured  CallRate        Q.2 P.11 P.12 P.22       Pexact Chromosome
rs10        1197 0.9622186 0.13700919  888  290   19 4.635677e-01          1
rs18        1189 0.9557878 0.28511354  605  490   94 7.759191e-01          1
rs29        1176 0.9453376 0.14285714  859  298   19 2.832575e-01          1
rs65        1185 0.9525723 0.72700422   83  481  621 4.647357e-01          1
rs73        1187 0.9541801 0.01053075 1167   15    5 3.988770e-08          1
rs114       1190 0.9565916 0.12184874  918  254   18 8.924018e-01          1
rs128       1183 0.9509646 0.02409129 1129   51    3 2.747904e-02          1
rs130       1188 0.9549839 0.68392256  117  517  554 8.407527e-01          1
rs143       1192 0.9581994 0.48489933  320  588  284 6.848365e-01          1
rs150       1182 0.9501608 0.66624365  127  535  520 5.568363e-01          1
```

in controls

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

```
      NoMeasured  CallRate        Q.2 P.11 P.12 P.22       Pexact Chromosome
rs10        1177 0.9453815 0.12744265  897  260   20 7.933317e-01          1
rs18        1185 0.9518072 0.27426160  623  474   88 9.418133e-01          1
rs29        1188 0.9542169 0.13215488  897  268   23 5.288436e-01          1
rs65        1183 0.9502008 0.71344041   98  482  603 8.871139e-01          1
rs73        1188 0.9542169 0.01641414 1154   29    5 6.941219e-06          1
rs114       1192 0.9574297 0.11157718  941  236   15 8.846527e-01          1
rs128       1197 0.9614458 0.02589808 1141   50    6 7.745807e-05          1
rs130       1181 0.9485944 0.70491109  104  489  588 8.887439e-01          1
rs143       1174 0.9429719 0.46805792  334  581  259 8.604122e-01          1
rs150       1176 0.9445783 0.65306122  139  538  499 7.968462e-01          1
```