GWAS Analysis in Stratified Populations - LUPA Workshop

Marcin Kierczak, Xia Shen, Katarina Tengvall,
Yurii Aulchenko, Örjan Carlborg

May 9, 2011

Introduction

To learn more about R visit: http://www.r-project.org

This short tutorial has been written to share our experience in the GWAS analysis of stratified data with all LUPA Community Members. Many of you encountered stratification in your datasets and realized that simple genomic control is not enough to solve the problem... Our Computational Genetics Group contributes to the LUPA project by developing new computational methods for the analysis of genomic data. Recently, we have been working on the problem of analyzing stratified populations quite a lot and came up with some solutions and guidelines. During this workshop we would like to share our approach with you.

We closely collaborate with Yurii Aulchenko's group and throughout this tutorial we will be using GenABEL R package developed by Yurii and his team. Conveniently, GenABEL is fully-integrated with R – an extremely powerful open-source software for statistical analysis. R is also very good in visualizing your data and it is definitely worth learning. In the context of GWAS analysis, the advantage is that many different models implemented in other GWAS analysis tools are available in this package. This makes GenABEL a perfect tool for comparing performance of different models: you load your data only once and can carry-on analysis using e.g., standard χ^2 test, genomic control, Eigenstrat approach, mixed model etc. During this workshop, we will show you how to do this on your data. Enjoy!

Authors

To consider

Below, we present various techniques and aspects of GWAS data analysis. You should, however, be aware that e.g., threshold values we used are not set in stone and we used them for demonstration purposes only. Every dataset is different and it has to be approached individually! We also reduced the size of the data in order to speed up computations.

Acknowledgements

We would like to thank Weronica Ek and Teresa Szczepińska for their comments on the manuscript of this tutorial. We are also grateful to Hille Fieten and Magnus Jansson for her help during the practicals.

0.1 Conventions used in this tutorial

Essential code chunks are marked by the red stripe.

Non-essential code and hints are marked by the blue stripe.

Quotations are marked by the black stripe.

Code is written in `Courier-like fonts`.

Code to be written in r console begins with >. Comments begin with # and #^ pertains to the line(s) immediately above the comment.

```
> print("Hello R user!")
[1] "Hello R user!"
> # ^ Prints "Hello R user"
>
> # Another print command
> print("Hello 2")
[1] "Hello 2"
```

What is R?

A Wikipedia entry describes R in the following words:

> R is a programming language and software environment for statistical computing and graphics. R is an implementation of the S programming language created by John Chambers while at Bell Labs combined with lexical scoping semantics inspired by Scheme. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is now developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors (Robert Gentleman and Ross Ihaka), and partly as a play on the name of S.
>
> The R language has become a de facto standard among statisticians for the development of statistical software, and is widely used for statistical software development and data analysis.
>
> R is part of the GNU project. Its source code is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. R uses a command line interface, though several graphical user interfaces are available.

Below, we provide some very basic introduction to R environment. However, you would greatly benefit from learning more about R from these sources:

- R project website.
- Peter Daalgard. Introductory Statistics with R. Springer, 2002.
- Intro to R.
- R Wikibook.
- Quick-R.

Working in R

Here we will briefly discuss the very basic aspects of working with R. The basic skills covered in this paragraph are absolutely necessary to work with R. If you are already familiar with R environment, you can skip this section.

Step 1 – installation.

To install R, follow instructions provided at the official website.

Step 2 – basic commands.

- To run R, simply double-click on the appropriate icon.
- You can quit R environment by typing `q()` and pressing Enter/Return. Beware that when you quit, all the data and results you worked with may be lost!
- You can use R as an ordinary calculator. Type e.g., `2 + 2 * 5`: and R will print the answer which, in this case, is `12`.
- You can see all the objects in your workspace by typing `ls()`.
- To get general help, type: `help.start()`. If you want to get help on a specific function, type either: `help(myfunction)` or `?myfunction`.

Step 3 – installing GenABEL.

Any R-package can be installed using `install.packages()` command, e.g.: if GenABEL is not installed, type: `install.packages("GenABEL")` and follow the instructions.

Chapter 1

Data Analysis

1.1 Loading Data

You are provided with two files:

- genotype information file: `genotype.raw`
- phenotype information file: `phenotype.dat` (you can open and edit it in a text editor)

First, we need to load GenABEL and load the available information into a data-structure that we will be working with. Type the following in R console:

```
> require("GenABEL")
GenABEL v. 1.6-6 (March 31, 2011) loaded

Installed GenABEL version (1.6-6) is not the same as stable
version available from CRAN (1.6-5). Unless used intentionally,
consider updating to the latest CRAN version. For that, use
'install.packages("GenABEL")', or ask your system administrator
to update the package.
> # ^ Load GenABEL
>
> # Define paths
> geno <- "data/genotype.raw"
> pheno <- "data/phenotype.dat"
> # Load data
> load.gwaa.data(pheno, geno, makemap=T) -> data
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!
increase in map order FORCED
```

GenABEL provides a wide range of conversion tools that let you import data stored in various file formats. For example, in order to load standard PLINK `*.ped` and `*.map` files, you can issue the following command:
`convert.snp.ped("genotype.ped", "genotype.map", "genotype.raw")`.
To get more information type:
`?convert.snp.ped`
and enjoy rich help.

1.2 Obtaining Basic Information About Data

Now your data have been loaded and are stored in the `data` object. This object consists of two main containers: one for genotype and one for phenotype. They can be accessed by two accessors: `gtdata` and `phdata` respectively. Our binary trait is named `bt` and the continuous trait is named `ct`. Type the code shown below to understand the GenABEL data structure:

```
> phdata(data)

> phdata(data)[2,]
           id sex bt       ct group response
dog225 dog225   1  0 1.925575     3 1.569402
> # ^ See phenos of the 2nd ind
>
> # See phenotypes of the dogs 3 to 7
> phdata(data)[3:7,]
           id sex bt       ct group  response
dog226 dog226   0  1 2.431597     3 0.7754572
dog227 dog227   1  0 4.175280     1 1.3674980
dog228 dog228   0  0 4.026864    NA 3.0488350
dog229 dog229   0  1 2.972239     1 1.2468811
dog230 dog230   1  0 2.017118     3 1.2822335
> # See sex of the first 5 individuals
> phdata(data)[1:5, "sex"]
[1] 1 1 0 1 0
> # See sex and the value of bt for these dogs
> phdata(data)[1:5,c("sex", "bt")]
       sex bt
dog224   1  0
dog225   1  0
dog226   0  1
dog227   1  0
dog228   0  0

> phdata(data)[,c("sex", "bt")]
```

By now you should understand how to access rows and columns in the phenotype data. Time for uncovering the representation of the genotype data. Be careful when typing – genomic data is huge and if you try to print too much, it may take a long time. In case it happens to you, do not hesitate to click the "STOP" button in the top-left corner of the terminal. OK, as we said – time for genotype data. Let us read genotype at markers 3 through 5 in the first individual:

```
> gtdata(data)[1,3:5]
@nids = 1
@nsnps = 3
@nbytes = 1
@idnames = dog224
@snpnames = BICF2P1383091 TIGRP2P259 BICF2P186608
@chromosome = 1 1 1
@coding =  04 01 01
@strand =  00 00 00
@map = 3212349 3249189 3265742
@male = 1
@gtps =
80 40 40
```

As you can see there are different slots, containing different type of information. Let's examine it more deeply. What is the chromosome for markers 1 through 3?

```
> chromosome(gtdata(data)[,1:3])
[1] "1" "1" "1"
> # And the name of the 17 000-th marker?
> snpnames(gtdata(data)[,17000])
[1] "BICF2P1431987"
> # What are the actual genotypes at marker 1177
> # for the first four dogs?
> as.character(gtdata(data)[1:4, 1177])
       BICF2G630712331
dog224 "T/T"
dog225 "T/T"
dog226 "T/G"
dog227 "T/T"
> # Reference allele at 1177 is apparently "T":
> refallele(gtdata(data)[,1177])
BICF2G630712331
            "T"
> # And the effective allele is "G":
> effallele(gtdata(data)[,1177])
BICF2G630712331
            "G"
```

You can also learn more about your markers by requesting summary of the data. In summary you will find *per marker* information about:

- chromosome,
- map position (chromosome-wise or genome-wise depending on whether you loaded the data with `makemap=T`),
- strand (u for unknown),
- allele coding,
- number of observed genotypes,
- call rate,
- allelic frequency,
- genotypic distrbution (counts),
- p-value of the exact test for HWE,
- Fmax (estimate of deviation from HWE),
- LRT p-value for HWE test are listed

Get summary for the first 5 individuals:

```
> summary(gtdata(data))[1:5,]
                Chromosome Position Strand A1 A2 NoMeasured
BICF2P1173580            1  3079928      u  G  A        207
BICF2G630707846          1  3082514      u  1  2        206
BICF2P1383091            1  3212349      u  A  G        207
TIGRP2P259               1  3249189      u  1  2        207
BICF2P186608             1  3265742      u  1  2        207
                CallRate Q.2 P.11 P.12 P.22       Pexact
BICF2P1173580   1.000000 0.5    0  207    0 1.240547e-61
BICF2G630707846 0.995169 0.0  206    0    0 1.000000e+00
BICF2P1383091   1.000000 0.5    0  207    0 1.240547e-61
TIGRP2P259      1.000000 0.0  207    0    0 1.000000e+00
BICF2P186608    1.000000 0.0  207    0    0 1.000000e+00
                Fmax         Plrt
BICF2P1173580     -1 2.282010e-64
BICF2G630707846    0 1.000000e+00
BICF2P1383091     -1 2.282010e-64
TIGRP2P259         0 1.000000e+00
BICF2P186608       0 1.000000e+00
```

If you need more details, use help: `?summary.snp.data` or `?snp.data-class`. In a similar way you can obtain summary for phenotype data. First, get summary for all the traits and co-variates:

```
> summary(phdata(data))
      id                 sex               bt
 Length:207         Min.   :0.0000   Min.   :0.0000
 Class :character   1st Qu.:0.0000   1st Qu.:0.0000
 Mode  :character   Median :0.0000   Median :0.0000
                    Mean   :0.4589   Mean   :0.3073
                    3rd Qu.:1.0000   3rd Qu.:1.0000
                    Max.   :1.0000   Max.   :1.0000
                                     NA's   :2.0000
       ct              group           response
 Min.   :0.1797   Min.   : 1.000   Min.   :-0.4325
 1st Qu.:2.8468   1st Qu.: 1.000   1st Qu.: 0.9996
 Median :3.7654   Median : 2.000   Median : 1.4090
 Mean   :3.7631   Mean   : 1.865   Mean   : 1.4859
 3rd Qu.:4.6989   3rd Qu.: 3.000   3rd Qu.: 1.9477
 Max.   :7.5664   Max.   : 3.000   Max.   : 3.3858
                  NA's   :66.000   NA's   : 1.0000
> # ^ Get summary of phenotype data
>
> # Get summary for continuous trait ct only.
> summary(phdata(data)["ct"])
       ct
 Min.   :0.1797
 1st Qu.:2.8468
 Median :3.7654
 Mean   :3.7631
 3rd Qu.:4.6989
 Max.   :7.5664
```

1.3 Preliminary Quality Control – QC1

Before we will be able to perform further steps of our analysis, we need to remove noisy and problematic data that might otherwise affect our GWAS results in an undesirable way. In GenABEL, there is a function called `check.marker` that is particularly suitable for our purposes. First, we recommend you to type the `?check.marker` command to learn a bit more about the possibilities the function gives. Preliminary quality control (QC1) will not be very strict – we do not want to get rid of any data that may contain true signal. We will treat cases and controls equally at this stage. Here, we have decided to use the following criteria:

- all the markers with call rate < 0.95 will be removed,
- all the individuals with more than 5% missing genotypes will be removed,
- all the markers with minor allele frequency $< 10^{-8}$ will be discarded,
- all the markers which are very strongly out of Hardy-Weinberg equilibrium (p-level $< 10^{-8}$) will be removed.

As you see, call rate is our main concern now. The `check.marker` function will also check whether there are any redundant (identical) individuals in the dataset.

Important note! You should **ALWAYS** consult the manual provided by your SNP chip manufacturer when setting QC thresholds.

Do quality control:

```
> qc1 <- check.marker(data, call = 0.95, perid.call = 0.95,
        maf = 1e-08, p.lev = 1e-08)
Excluding people/markers with extremely low call rate...
174375 markers and 207 people in total
0 people excluded because of call rate < 0.1
1069 markers excluded because of call rate < 0.1
Passed: 173306 markers and 207 people

RUN 1
173306 markers and 207 people in total
42743 (24.66331%) markers excluded as having low (<1e-06%) minor allele frequency
1468 (0.8470567%) markers excluded because of low (<95%) call rate
2106 (1.215192%) markers excluded because they are out of HWE (P <1e-08)
0 (0%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2663605 (s.e. 0.01925083)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7793032 (s.e. 0.01515045), as based on 2000 autosomal markers
2 (0.9661836%) people excluded because of too high IBS (>=0.95)
In total, 127614 (73.63507%) markers passed all criteria
In total, 205 (99.03382%) people passed all criteria

RUN 2
127614 markers and 205 people in total
0 (0%) markers excluded as having low (<1e-06%) minor allele frequency
0 (0%) markers excluded because of low (<95%) call rate
5 (0.003918065%) markers excluded because they are out of HWE (P <1e-08)
0 (0%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2665348 (s.e. 0.01926229)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.77931 (s.e. 0.01463823), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 127609 (99.99608%) markers passed all criteria
In total, 205 (100%) people passed all criteria

RUN 3
127609 markers and 205 people in total
0 (0%) markers excluded as having low (<1e-06%) minor allele frequency
0 (0%) markers excluded because of low (<95%) call rate
0 (0%) markers excluded because they are out of HWE (P <1e-08)
0 (0%) people excluded because of low (<95%) call rate
Mean autosomal HET is 0.2665348 (s.e. 0.01926229)
```

```
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7771738 (s.e. 0.01443821), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 127609 (100%) markers passed all criteria
In total, 205 (100%) people passed all criteria
```

Note that the QC procedure is iterative: at the first round, some individuals and markers are removed and values such as departure from HWE have to be re-computed. In our example, all individuals and markers that passed iteration 2 passed also iteration 3 so the process finished.

The results of the QC1 have been stored as the `qc1` object. If you need more details, you can type the `summary(qc1)` command. Now we need to create a new dataset that contains only the markers and individuals that passed QC1. This is how it is done:

```
> data.qc1 <- data[qc1$idok, qc1$snpok]
```

From now on `data.qc1` is the dataset we will work with.

Hint: if you are wondering which individuals have been excluded during QC1, please write:

```
> excluded_dogs_qc1 <- setdiff(phdata(data)$id,idnames(data.qc1))
> excluded_dogs_qc1
[1] "dog242" "dog297"
```

Now, that we have cleaned the data, we can proceed towards more advanced trait and covariate analysis and visualization.

1.4 Trait/Covariate Analysis and Visualization

It is time to examine potential co-variates to be included in our models. Let us begin with checking whether there is any significant difference between the number of females and males between cases and controls. First we will need to tabularize our data, next we will use the constructed table as input to the Fisher's exact test for count data.

```
> tab <- table(phdata(data.qc1)$bt, phdata(data.qc1)$sex)
> # ^ displays table: rows - bt, columns - sex
>
> # Do Fisher's exact test
> fisher.test(tab)
        Fisher's Exact Test for Count Data

data:  tab
p-value = 1
```

```
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.5327872 1.9186142
sample estimates:
odds ratio
  1.012806
```

As you see, there is no significant difference here. Sex was an example of a discrete variable, let us examine a continuous variable, **response**. First, we will visualize the difference in response between cases and controls using box-and-whisker plot. The bottom and top of the box are the 25th and 75th percentile (the lower and upper quartiles) respectively. The thicker band near the middle of the box is the 50th percentile (the median). The ends of the whiskers represent the lowest datum still within 1.5 IQR of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile. Circles represent outliers.

```
> boxplot(phdata(data.qc1)$response ~ phdata(data.qc1)$bt,
    names=c("Controls","Cases"), ylab="Response")
```

The above code will produce a nice box-and-whisker plot as presented in Figure 1.1.

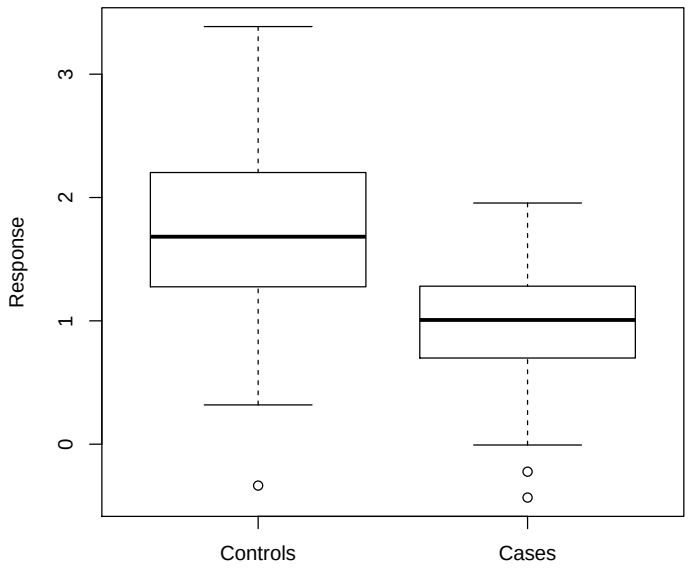

Figure 1.1: box-and-whisker visualization of response in controls and cases.

Always remember to close your graphical device using the `dev.off()` function!

Hint: it is nice to watch plots on screen, but we often need to save them as pictures. Since taking screenshots is not the best way of generating publication-quality figures, R provides a multitude of the so-called graphical devices. A device can be simply seen as a workshop that produces some kind of graphical output, e.g.: **jpeg**, **png**, **bitmap** or **pdf**. Screen is also a graphical device and in R it is the default one. So, in order to generate, say, a pdf, we need to:

1. start a new PDF graphical device: `pdf(file="myimage.pdf")`
2. plot our stuff: `hist(rnorm(100,0,1))`
3. close the device: `dev.off()`

It is actually **very important** to close graphical devices. If you open a new device without closing the old one, there can be a true mess. To summarize, in order to save the box-and-whisker plot that we have made for Figure 1.1 here, we wrote:

```
> pdf(file="tutorial/images/boxplotResponse.pdf")
> boxplot(phdata(data.qc1)$response ~ phdata(data.qc1)$bt,
    names=c("Controls","Cases"), ylab="Response")
> dev.off()
```

Now, back to the analysis. From the box-and-whisker plot, we can see that there is a difference in **response** between cases and controls. Is this difference significant? We can use Welch two sample t-test to determine this:

```
> t.test(phdata(data.qc1)$response ~ phdata(data.qc1)$bt)
        Welch Two Sample t-test

data:  phdata(data.qc1)$response by phdata(data.qc1)$bt
t = 8.608, df = 155.905, p-value = 7.603e-15
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.5677410 0.9058987
sample estimates:
mean in group 0 mean in group 1
      1.7207394       0.9839196
```

The difference is significant at our significance level (p-value < 0.05), it may be a good idea to treat **response** as a covariate/fixed effect.
In a case where we have two continuous random variables, such as **ct** and **response** in our data, we can see whether there is any relationship between them by means of simple visualization:

```
> plot(phdata(data.qc1)$ct, phdata(data.qc1)$response,
    pch=19, cex=.9, xlab="Continuous trait", ylab="Response")
```

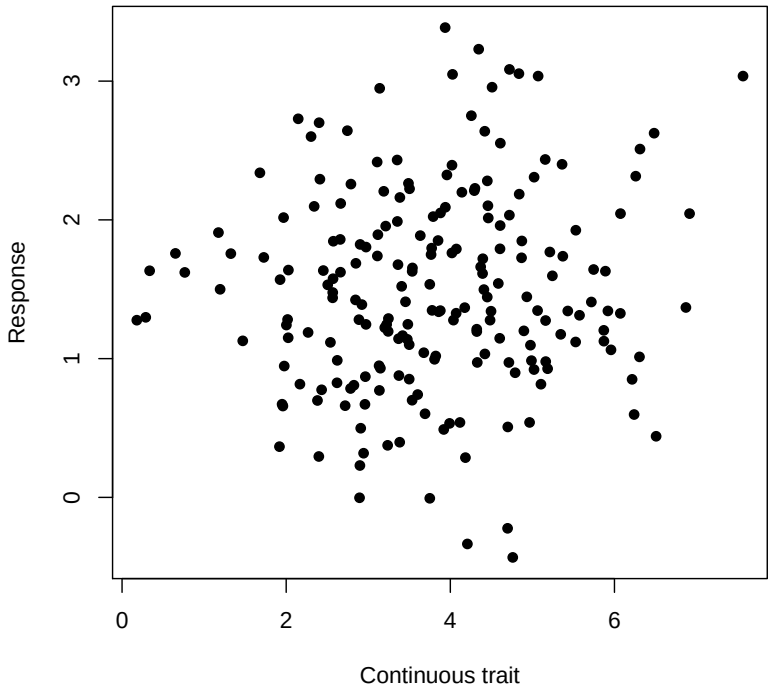

Figure 1.2: Visualization of **ct** vs. **response**.

From Figure 1.2, it seems there is no correlation between **ct** and **response**, but we will perform Pearson's correlation test to be sure:

```
> cor.test(phdata(data.qc1)$ct, phdata(data.qc1)$response)
        Pearson's product-moment correlation

data:  phdata(data.qc1)$ct and phdata(data.qc1)$response
t = 1.1893, df = 202, p-value = 0.2357
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.0546059  0.2182617
sample estimates:
       cor
0.08339082
```

Indeed, we cannot reject our null hypothesis $H_0 : cor = 0$.

Hint: note how we tweaked R plot (Figure 1.2). We made data points slightly (by 10%) smaller by using `cex=.9` and we changed empty circle to a filled point by using `pch=19`. The `pch` parameter is a bit cryptic. Below (Figure 1.3) you have all R `pch` symbols with their codes.

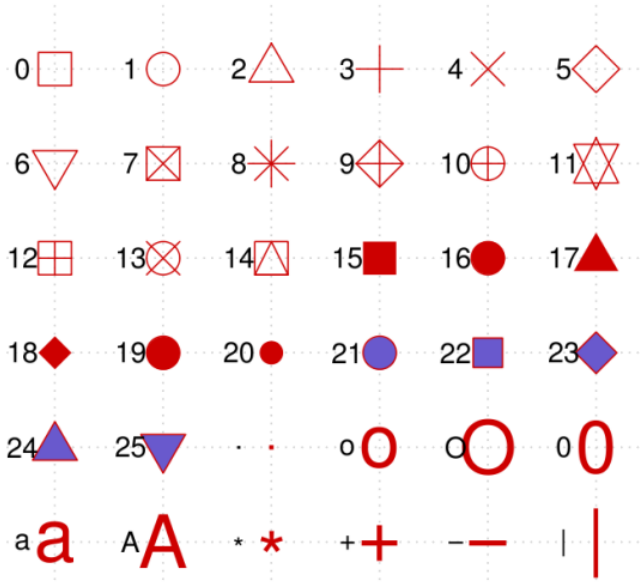

Figure 1.3: pch codes. After: http://rgraphics.limnology.wisc.edu.

Once we know which variables should be included in the analysis, we can proceed towards analyzing population structure.

1.5 Analysis of Population Structure

Population structure can be determined in several different ways, based on what type of data is available etc. Here, we will focus on using only genomic data. First, we need a genomic kinship matrix that shows how closely the individuals are related to each other. The genomic kinship matrix is computed using only genotype information. To simplify the situation, we will build our matrix using autosomal chromosomes only. We know that the X chromosome has been assigned number 39 and we will use this fact to determine autosomal markers. If chromosome '39' was called 'X', it would be automatically recognized!

```
> autosomalMarkers <- which(chromosome(data.qc1) != "39")
> # ^ Determine autosomal markers (their ids)
>
> # Check how many markers are autosomal
> length(autosomalMarkers)
[1] 125209
```

```
> # Now we need the actual names of the autosomal markers
> autosomalMarkerNames <- snpnames(data.qc1)[autosomalMarkers]
```

Now we can compute genomic kinship matrix for our dataset. Following this, we will transform it to a distance matrix and, finally, visualize using multidimensional scaling (MDS).

```
> data.qc1.gkin <- ibs(data.qc1[,autosomalMarkerNames], weight="freq")
> # ^ Compute genomic-kinship matrix using autosomal markers
>
> # Compute the distances
> data.qc1.dist <- as.dist(0.5 - data.qc1.gkin)
> # Check how are the distances distributed. Use 100 breaks, just for fun :-)
> hist(data.qc1.dist, breaks=100)
> # Do multidimensional scaling
> data.qc1.mds <- cmdscale(data.qc1.dist)
> # Plot the results
> plot(data.qc1.mds, pch=19, cex=.5, xlab="MDS1", ylab="MDS2")
```

It is really interesting! It looks like we have somehow more dense group to the left and more dispersed group to the right of the plot. Stratification? We will investigate it shortly, but now in the right-bottom part of the plot, there are two dogs that appear to be outliers. Let's try to identify them. R has a very nice function `identify()` that will help us to do this.

```
> plot(data.qc1.mds, pch=19, cex=.5, xlab="MDS1", ylab="MDS2")
> # ^ Plot MDS results
>
> # Identify the outliers by clicking on them.
> #Once you are done right-click and close the plot window.
>
> # IMPORTANT! If you want to identify using mouse,
> #uncomment the first line and comment out the second.
>
> ######## MODIFY THIS DEAR USER ###########
> # outliers <- identify(data.qc1.mds)
> outliers <- c(50, 96)
> ##########################################
>
> # Plot window is closed. Back to the console.
> outliers
[1] 50 96
> # Get outlier and non-outlier names
> outlierNames <- phdata(data.qc1)[outliers,]$id
> nonOutlierNames <- phdata(data.qc1)[-outliers,]$id
> outlierNames
[1] "dog274" "dog321"
```

```
> # Store outlier coordinates
> outliersX <- data.qc1.mds[outliers,1]
> outliersY <- data.qc1.mds[outliers,2]
> save.image("page15.dat")
```

Probably it is a good idea to have a closer look at the dog274 and dog321. Are they really outliers? Here, we will assume that they are.

So far, we have been drawing only very basic plots. But maybe our stratified population deserves a better visualization? We can visualize many things on our MDS-plot: sex, case/control etc.

```
> ctrls <- which(phdata(data.qc1)$bt == 0)
> ctrlsX <- data.qc1.mds[ctrls,1]
> ctrlsY <- data.qc1.mds[ctrls,2]
> cases <- which(phdata(data.qc1)$bt == 1)
> casesX <- data.qc1.mds[cases,1]
> casesY <- data.qc1.mds[cases,2]
> # ^ Gather data, get coordinates for controls and cases.
>
> # Make a plot with outliers
> plot(data.qc1.mds, pch=19, cex=.5, xlab = "MDS 1",
    ylab = "MDS 2", main="Outliers ")
> text(outliersX + 0.03 , outliersY, labels=outlierNames)
> points(outliersX, outliersY, pch=19, cex=.5, col="red")
> #If you want names/ids of the cases
> casesNames <- phdata(data.qc1)[cases,]$id
> casesIds <- as.numeric(idnames(data.qc1) %in% casesNames)
```

You can see the result in Figure 1.4. Note that we added 0.03 to `outliersX` coordinate in order to avoid the text overlapping the datapoints.

Now let us do some more "sophisticated" plot. Say we want to visualize **sex** and **bt** on the same plot using different `pch` symbols. We have decided to use symbols as summarized below:

	Female (0)	Male (1)
Control (0)	circle (`pch=1`)	square (`pch=0`)
Case (1)	crossed circle (`pch=13`)	crossed square (`pch=12`)

Let us do this:

```
> sex <- phdata(data.qc1)$sex
> females <- which(sex == 0)
> males <- which(sex == 1)
> # Symbol will contain the appropriate pch value
> symbol <- sex
> # Assume every individual is a control
> symbol[males] <- 0
```

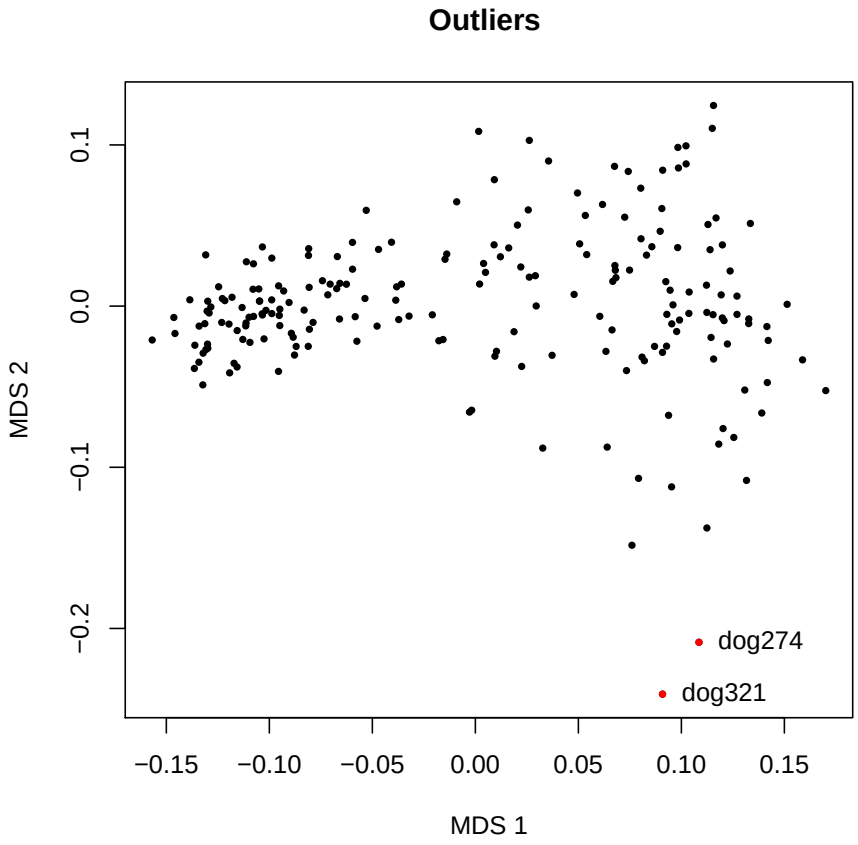

Figure 1.4: Visualization of MDS performed on the genomic-kinship matrix. The identified outliers are represented as red dots.

```
> symbol[females] <- 1
> # Take care of the cases
> femaleCases <- which(females %in% cases)
> maleCases <- which(males %in% cases)
> symbol[femaleCases] <- 13
> symbol[maleCases] <- 12
> # Et voila!
> plot(data.qc1.mds, pch=symbol, main="Cases/controls",
    col="red", xlab="MDS 1", ylab="MDS 2", cex=1.2)
```

The result you can enjoy by looking at Figure 1.5.

Coming back to our main analysis – now we will remove the outliers from the dataset.

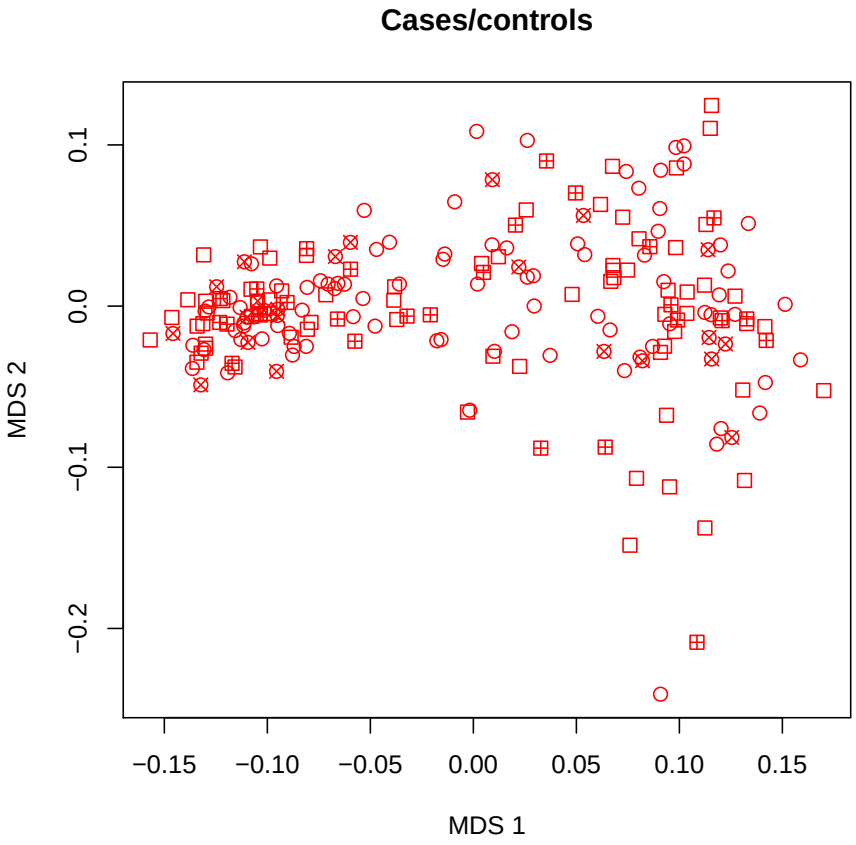

Figure 1.5: A more complex visualization of MDS, **sex** and **bt**.

```
> data2 <- data.qc1[nonOutlierNames, ]
> # ^ Create a new dataset that does not contain outliers
>
> # Check if everything is ok
> nids(data2)
[1] 203
> save.image("page18.dat")
```

We are ready to perform the final quality control (QC2).

1.6 Final Quality Control – QC2

The second QC will be more strict and we will take care of the markers that are out of Hardy-Weinberg equilibrium. However, in cases, departure from HWE

may be due to the actual signal. Hence, we will remove only the markers which are out of HWE in controls. The `fdrate` parameter sets false-discovery rate threshold for determining out-of-HWE markers.

```
> qc2 <- check.marker(data2, hweids = (phdata(data2)$bt == 0),
    call=0.95, perid.call=0.95, fdrate = 0.2)
Excluding people/markers with extremely low call rate...
127609 markers and 203 people in total
0 people excluded because of call rate < 0.1
0 markers excluded because of call rate < 0.1
Passed: 127609 markers and 203 people

RUN 1
127609 markers and 203 people in total
12676 (9.933469%) markers excluded as having low (<1.231527%) minor allele frequency
0 (0%) markers excluded because of low (<95%) call rate
1157 (0.9066759%) 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 0.2956728 (s.e. 0.02080814)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7548282 (s.e. 0.01536849), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 113778 (89.16142%) markers passed all criteria
In total, 203 (100%) people passed all criteria

RUN 2
113778 markers and 203 people in total
0 (0%) markers excluded as having low (<1.231527%) 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 0.2956728 (s.e. 0.02080814)
0 people excluded because too high autosomal heterozygosity (FDR <1%)
Mean IBS is 0.7570448 (s.e. 0.01530842), as based on 2000 autosomal markers
0 (0%) people excluded because of too high IBS (>=0.95)
In total, 113778 (100%) markers passed all criteria
In total, 203 (100%) people passed all criteria
> # ^ Perform QC2, remove markers that are out of HWE in controls only!"
>
> # Create our final dataset
> data.clean <- data2[qc2$idok, qc2$snpok]
> # How many markers passed QC2
> nsnps(data.clean)
[1] 113778
> # Determine controls and cases
> ctrls <- which(phdata(data.clean)$bt == 0)
> cases <- which(phdata(data.clean)$bt == 1)
> # Check HWE departure separately for controls (accessing only the second element
> descriptives.marker(data.clean, idsubset=ctrls)[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 1155.00 5523.000 113778
Prop        0        0    0.01    0.049      1
> # And for cases
> descriptives.marker(data.clean, idsubset=cases)[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  172.000 1062.000 4294.000 113778
Prop        0    0.002    0.009    0.038      1
```

What about performing preliminary association test on our cleaned data?

1.7 Preliminary Test for Association

Let us perform the first simple association test:

```
> an0 <- qtscore(bt ~ response, data.clean, trait="binomial")
```

Analysis is ready. We can check inflation factor λ and draw Q-Q plot before and after the Genomic Control (GC) correction (c.f. Figure 1.6).

```
> lambda(an0)
$estimate
[1] 1.309801

$se
[1] 4.330022e-05

$iz0
[1] 1.297896

$iz2
[1] 1.08766
> # ^ Check inflation factor lambda
>
> # Plot standard Q-Q plot
> estlambda(an0[,"P1df"])
$estimate
[1] 1.309801

$se
[1] 4.330022e-05
> # And the same Q-Q plot, after the genomic control (GC) correction
> estlambda(an0[,"Pc1df"])
```

```
$estimate
[1] 1

$se
[1] 3.305863e-05
```

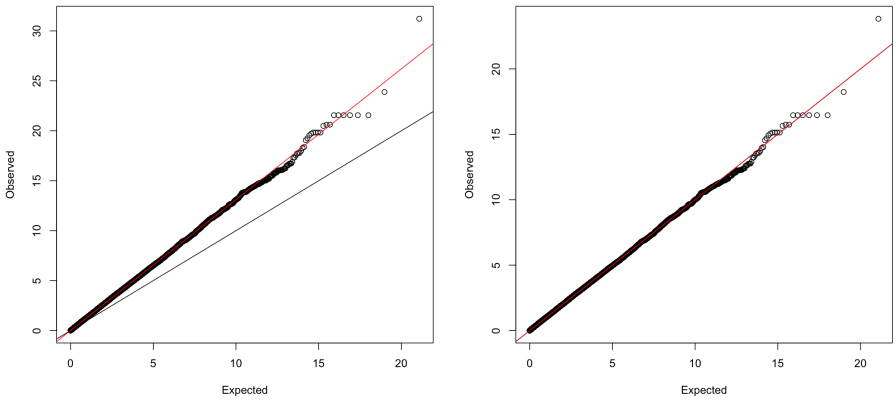

Figure 1.6: Q-Q plots before (left) and after (right) GC control. Both plots show Actual vs. expected χ values.

Use `png()` or `jpeg()` for storing plots with many data points, such as Manhattan plots. This will save a lot of disk space!

If you want to save your Q-Q plot into a file, it is probably wise to use *.png. In PDF format, all the points on the plot will take a lot of disk space. Below we show how to set PNG device to plot 2 Q-Q plots side-by-side at resolution 155dpi.

```
> png(file = "Desktop/LUPA Workshop/tutorial/images/an0qq.png",
        width = 35, height = 17.5, units = "cm", res = 155)
> par(mfcol = c(1, 2))
> estlambda(an0[, "P1df"])
> estlambda(an0[, "Pc1df"])
> dev.off()
```

1.8 Determining Number of Subpopulations

By now, we have a clean dataset, and we know that there is pretty strong stratification in the data. By looking at Figure 1.4, we can guess there are two subpopulations, but while intuition is often a good guide, statistical test is even better... To determine the optimal number of subpopulations (clusters), we will use an iterative K-means clustering. In K-means clustering it is the user who sets the number of clusters and the algorithm assigns individuals to clusters.

We can run a series of K-means clusterings with different number K of clusters and see which setting is optimal. We can judge the goodness of a clustering by looking at within-cluster sums of squares (WSS). By minimizing WSS, we pick up the optimal K. So, our goal is now to have a plot with "average WSS vs. K". Let's do this:

```
> autosomalMarkers <- which(chromosome(data.clean) != "39")
> autosomalMarkerNames <- snpnames(data.clean)[autosomalMarkers]
> data.clean.gkin <- ibs(data.clean[,autosomalMarkerNames], weight="freq")
> data.clean.dist <- as.dist(0.5 - data.clean.gkin)
> data.clean.mds <- cmdscale(data.clean.dist)
> # ^ Re-compute genomic kinship matrix for cleaned dataset
>
> plot(data.clean.mds, pch=19, cex=.5)
```

Plot WSS vs. K to determine number of clusters - look for the bend in the curve:

```
> wss <- (nrow(data.clean.mds) - 1) * sum(apply(data.clean.mds, 2, var))
> for (i in 2:15) wss[i] <- sum(kmeans(data.clean.mds,
    centers=i, nstart=nids(data.clean))$withinss)
> plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="WSS")
```

The result of iterative clustering is presented in Figure 1.7. You should look for a bend in the plot and choose the corresponding K. Here, the bend begins at $K = 2$ and is the strongest until $K = 4$. We will use $K = 2$, but one may also wish to test $K = 3$ and $K = 4$. Below, we will perform a number of clusterings equal to the number of individuals. Every time, cluster centers will be assigned randomly. The final result is computed by taking into account all these clusterings. Such approach minimizes potential errors. Following the clustering, we will create two vectors, each containing individuals assigned to a particular subpopulation. We will also store the coordinates, separately for individuals in each subpopulation. Finally we will plot our population structure on the MDS plot. We will add a simple legend. You should be able to understand the `legend()` syntax, just check (`?legend`) what is the meaning of the `pty` argument :-). The result you can see in Figure 1.8.

```
> km <- kmeans(data.clean.mds, centers=2, nstart=nids(data.clean))
> # ^ K-means clustering with K=2 as determined using WSS
>
> # Determine clusters and coordinates of individuals in each cluster
> cl1 <- which(km$cluster == 1)
> cl1num <- as.numeric(cl1)
> cl1x <- data.clean.mds[cl1num,1]
> cl1y <- data.clean.mds[cl1num,2]
> cl2 <- which(km$cluster == 2)
> cl2num <- as.numeric(cl2)
```

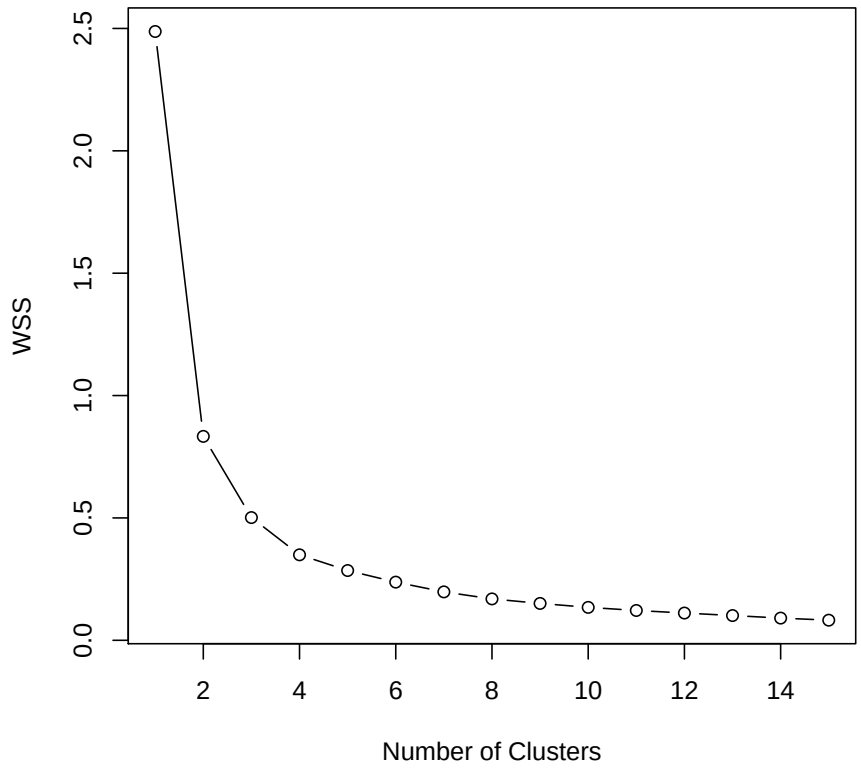

Figure 1.7: Within group sum of squares (WSS) as a function of the number of clusters (K). Determined using K-means clustering. It is probably good to further test K between 2 and 4.

```
> cl2x <- data.clean.mds[cl2num,1]
> cl2y <- data.clean.mds[cl2num,2]
> save.image("page22.dat")
> plot(data.clean.mds,type="n", xlab="MDS1",
    ylab="MDS2", main="Subpopulations (K=2)")
> points(cl1x, cl1y, pch=19, cex=.5, col="dodgerblue")
> points(cl2x, cl2y, pch=19, cex=.5, col="red")
> legend(x="topright", c("subpop 1", "subpop 2"),
    col=c("dodgerblue","red"), pch=c(19,19), ncol=1,
    bty="n", pt.cex=c(0.5,0.5))
```

Before we can perform association tests that take into account the structure of our population, we have to create a vector that represents the structure: each individual is assigned an id – the number of the subpopulation it belongs to.

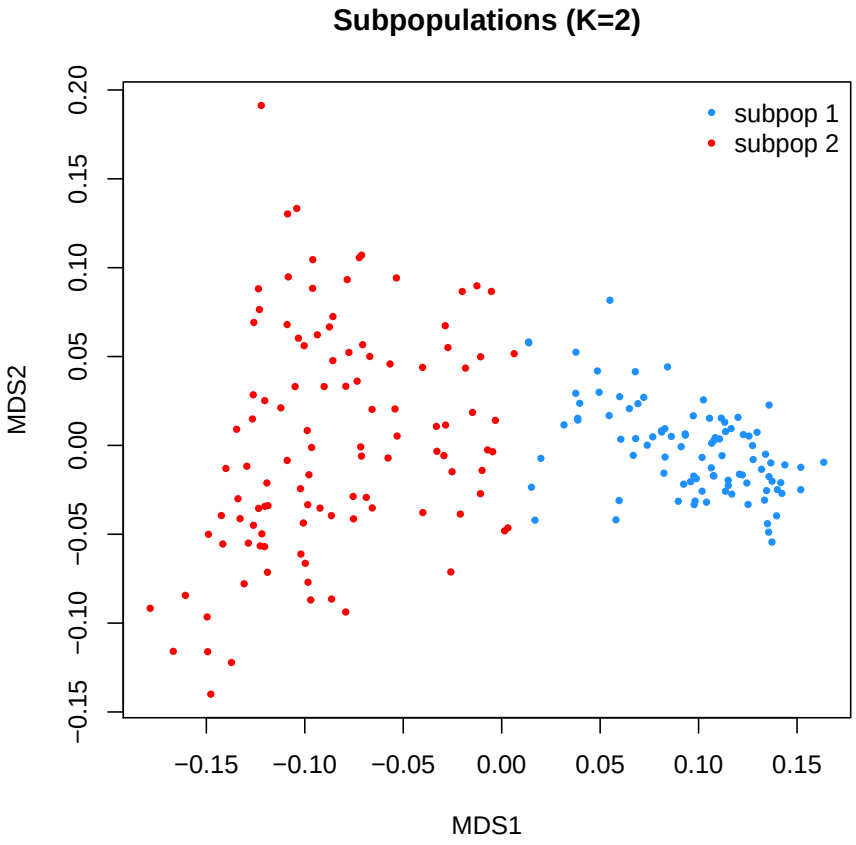

Figure 1.8: MDS plot with two subpopulations visualized using blue and red dots.

```
> cl1names <- names(which(km$cluster == 1))
> pop <- as.numeric(idnames(data.clean) %in% cl1names)
> pop
  [1] 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 0 1 0
 [28] 1 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 1
 [55] 1 0 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0
 [82] 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 1 1
[109] 1 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 1 0 0 0 1 0 0 1 1 0
[136] 0 0 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 1 1 0 0 1 0 1 1 0 1
[163] 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 0 0 0 0 1 0 0
[190] 0 0 0 1 1 1 0 0 0 1 0 0 0 0
```

Hint: you may want to visualize a continuous variable, say **response** on your MDS plot using different color intensity. In order to do this, you can use package `wasim`. First, you need to install it: `install.packages("wasim")`. Then you can proceed:

```
> require("wasim")
> levels <- phdata(data.clean)$response
> # We do not like levels to be negative
> levels <- levels + 1
> cl1cols <- color.factor("navy", levels, max=max(levels, na.rm=TRUE))
> cl2cols <- color.factor("red", levels, max=max(levels, na.rm=TRUE))
> plot(data.clean.mds, pch=19, cex=.8, col=cl1cols, xlab="MDS 1",
    ylab = "MDS 2", main = "Response intensity")
> points(cl2x,cl2y,pch=19, cex=.8, col=cl2cols)
```

You can see the result in Figure 1.9.

1.9 Association Tests in Stratified Populations

Now, we are ready to test different approaches to analyzing associations in stratified populations. GenABEL provides models summarized in Table 1.1:

Approach	Software	GenABEL function
Genomic control	PLINK	`qtscore(GenABEL)`
Stratified association	–	`*score(GenABEL)`
PCA	Eigenstrat	`egscore(GenABEL)`
Mixed model	EMMA	`mmscore(GenABEL)`
	EMMA-X	`mmscore(GenABEL)`

Table 1.1: Different approaches to correcting for stratification effect implemented in GenABEL.

Let us begin with the Structured Association (SA) approach. First, we will run association test taking into account population structure. Subsequently, we will look at the genomic inflation factor λ, draw a Q-Q plot, a Manhattan plot and we will list the most important markers.

```
> an.sa <- qtscore(bt ~ response, data.clean, strata=pop, trait="binomial")
> # ^ Run association tests taking strata into account
>
> # Estimate lambda
> lambda(an.sa)
$estimate
[1] 1.006821

$se
[1] 4.542152e-05
```

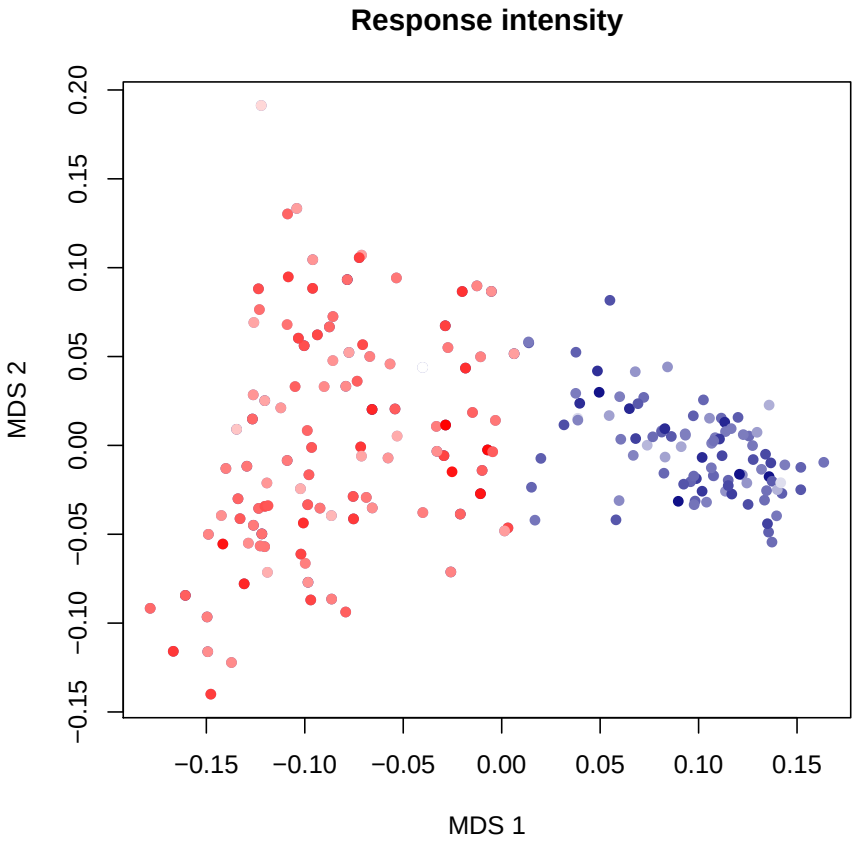

Figure 1.9: MDS plot with **response** visualized as color intensity separately for each subpopulation.

```
$iz0
[1] 1.028777

$iz2
[1] 1
> # Plot Q-Q plot
> estlambda(an.sa[,"P1df"])
$estimate
[1] 1.006821

$se
[1] 4.542152e-05
> # Plot standard Manhattan plot
> plot(an.sa, pch=19, cex=.5)
> # List top markers
```

```
> summary(an.sa)
Summary for top 10 results, sorted by P1df
                Chromosome   Position Strand A1 A2   N
BICF2P1132186           24 2322198067      u  T  C 200
BICF2P1330344           25 2389725054      u  C  G 200
BICF2S23629051          24 2323059128      u  T  C 200
TIGRP2P325071           25 2370626688      u  T  C 200
BICF2P157056            25 2370636198      u  T  C 200
TIGRP2P325076           25 2370647761      u  A  G 200
BICF2P389757            25 2370661070      u  T  G 200
TIGRP2P325085           25 2370676919      u  C  T 200
BICF2P1371165           25 2370742763      u  T  G 200
BICF2G630709860          1   10937977      u  G  A 200
                     effB   se_effB chi2.1df         P1df
BICF2P1132186   1.248829 0.2502058 24.91214 6.000339e-07
BICF2P1330344   1.664289 0.3687434 20.37085 6.379425e-06
BICF2S23629051  1.364165 0.3189909 18.28846 1.898540e-05
TIGRP2P325071   2.369053 0.5569040 18.09627 2.100127e-05
BICF2P157056    2.369053 0.5569040 18.09627 2.100127e-05
TIGRP2P325076   2.369053 0.5569040 18.09627 2.100127e-05
BICF2P389757    2.369053 0.5569040 18.09627 2.100127e-05
TIGRP2P325085   2.369053 0.5569040 18.09627 2.100127e-05
BICF2P1371165   2.369053 0.5569040 18.09627 2.100127e-05
BICF2G630709860 1.941192 0.4653461 17.40142 3.025993e-05
                    effAB    effBB chi2.2df         P2df
BICF2P1132186   1.168745 1.985456 29.43226 4.063170e-07
BICF2P1330344   1.700574       NA 20.37085 6.379425e-06
BICF2S23629051  1.445520       NA 18.28846 1.898540e-05
TIGRP2P325071   2.645360       NA 18.09627 2.100127e-05
BICF2P157056    2.645360       NA 18.09627 2.100127e-05
TIGRP2P325076   2.645360       NA 18.09627 2.100127e-05
BICF2P389757    2.645360       NA 18.09627 2.100127e-05
TIGRP2P325085   2.645360       NA 18.09627 2.100127e-05
BICF2P1371165   2.645360       NA 18.09627 2.100127e-05
BICF2G630709860 1.902534       NA 17.40142 3.025993e-05
                      Pc1df
BICF2P1132186   6.549350e-07
BICF2P1330344   6.856540e-06
BICF2S23629051  2.026138e-05
TIGRP2P325071   2.239809e-05
BICF2P157056    2.239809e-05
TIGRP2P325076   2.239809e-05
BICF2P389757    2.239809e-05
TIGRP2P325085   2.239809e-05
BICF2P1371165   2.239809e-05
BICF2G630709860 3.219638e-05
```

Similarly, we can perform our analysis using the PCA approach. The idea of this test is to use genomic kinship matrix to:

- derive axes of genetic variation (principal components),
- adjust both trait and genotypes using projection onto these axes.

Note that the diagonal of the kinship matrix should be replaced (default it is .5+F, and for EIGENSTRAT one needs variance). These variances are produced by `hom` function. By default, 3 first components are user for correction. However, you may specify the number of components by using `naxes` parameter. How to determine the number of components to use? This is a bit beyond the scope of this tutorial, but you can read about different criteria (e.g. *eigenvalue* ≥ 1 or scree test) here (SAS manual). Note, the default number of components is rather reasonable in the majority of the cases.

```
> diag(data.clean.gkin) <- hom(data.clean[,autosomalMarkerNames])$Var
> # ^ Replace the diagonal with right elements
>
> # Perform association tests
> an.eigen <- egscore(bt ~ response, data.clean, kinship=data.clean.gkin)
> # Check inflation
> lambda(an.eigen)
$estimate
[1] 1.048826

$se
[1] 5.707759e-05
> # Plot Q-Q plot
> estlambda(an.eigen[, "P1df"])
$estimate
[1] 1.048826

$se
[1] 5.707759e-05
> # Plot Manhattan
> plot(an.eigen)
> # Get summary
> summary(an.eigen)
Summary for top 10 results, sorted by P1df
               Chromosome   Position Strand A1 A2   N
BICF2P1132186          24 2322198067      u  T  C 200
BICF2P217273           24 2322356353      u  T  C 199
BICF2P189945           24 2322224208      u  G  A 200
BICF2P942043           24 2322435729      u  A  G 200
BICF2P1496251          24 2322340126      u  C  G 200
BICF2S23020296         25 2395662752      u  G  A 200
BICF2P1443008          25 2395638475      u  G  A 200
BICF2P1201324          25 2395647179      u  C  T 200
BICF2P471722           25 2395687081      u  G  A 200
BICF2P1047799          25 2395707310      u  T  C 200
                     effB     se_effB chi2.1df         P1df
```

```
BICF2P1132186  0.04507027 0.008631454 27.26544 1.773533e-07
BICF2P217273   0.04927724 0.010580527 21.69093 3.203015e-06
BICF2P189945   0.04521763 0.009884934 20.92513 4.775888e-06
BICF2P942043   0.04505385 0.009959706 20.46307 6.079291e-06
BICF2P1496251  0.06504030 0.015086561 18.58594 1.624141e-05
BICF2S23020296 0.04122949 0.009688519 18.10927 2.085830e-05
BICF2P1443008  0.02887591 0.007133887 16.38396 5.172096e-05
BICF2P1201324  0.02887591 0.007133887 16.38396 5.172096e-05
BICF2P471722   0.02887591 0.007133887 16.38396 5.172096e-05
BICF2P1047799  0.02887591 0.007133887 16.38396 5.172096e-05
                      Pc1df      effAB         effBB
BICF2P1132186  3.420989e-07 0.04507027  9.014055e-02
BICF2P217273   5.424729e-06 0.16872233 -6.185174e-04
BICF2P189945   7.945246e-06 0.04521763  9.043527e-02
BICF2P942043   1.000506e-05 0.04505385  9.010771e-02
BICF2P1496251  2.558266e-05 0.21255190  9.714451e-19
BICF2S23020296 3.249099e-05 0.04122949  8.245898e-02
BICF2P1443008  7.738048e-05 0.02887591  5.775183e-02
BICF2P1201324  7.738048e-05 0.02887591  5.775183e-02
BICF2P471722   7.738048e-05 0.02887591  5.775183e-02
BICF2P1047799  7.738048e-05 0.02887591  5.775183e-02
               chi2.2df P2df
BICF2P1132186        NA   NA
BICF2P217273         NA   NA
BICF2P189945         NA   NA
BICF2P942043         NA   NA
BICF2P1496251        NA   NA
BICF2S23020296       NA   NA
BICF2P1443008        NA   NA
BICF2P1201324        NA   NA
BICF2P471722         NA   NA
BICF2P1047799        NA   NA
```

Finally, let us use the mixed model approach. This is a bit more complex task – first we have to estimate polygenic model:

```
> h2a <- polygenic(bt ~ response, data.clean,
    kin = data.clean.gkin, trait="binomial")


> an.mm <- mmscore(h2a, data.clean)
> # ^ Perform association tests using mixed-model
>
> # Check inflation
> lambda(an.mm)
$estimate
[1] 1

$se
[1] NA
```

```
> # Get summary
> summary(an.mm)
Summary for top 10 results, sorted by P1df
               Chromosome   Position Strand A1 A2   N
BICF2P1132186          24 2322198067      u  T  C 200
BICF2P942043           24 2322435729      u  A  G 200
BICF2P189945           24 2322224208      u  G  A 200
BICF2P217273           24 2322356353      u  T  C 199
BICF2P1496251          24 2322340126      u  C  G 200
BICF2S23629051         24 2323059128      u  T  C 200
BICF2S23621851         24 2322970350      u  T  C 200
BICF2P1330344          25 2389725054      u  C  G 200
BICF2G63099370         25 2389061691      u  C  T 199
TIGRP2P325071          25 2370626688      u  T  C 200
                     effB    se_effB chi2.1df         P1df
BICF2P1132186  0.2343983 0.04460116 27.61956 1.476782e-07
BICF2P942043   0.2107946 0.04695219 20.15613 7.137105e-06
BICF2P189945   0.2101299 0.04684843 20.11803 7.280700e-06
BICF2P217273   0.2122052 0.04740729 20.03650 7.597807e-06
BICF2P1496251  0.2603582 0.05853679 19.78266 8.676625e-06
BICF2S23629051 0.2962620 0.07011152 17.85554 2.383242e-05
BICF2S23621851 0.2642291 0.06323119 17.46219 2.930795e-05
BICF2P1330344  0.4185938 0.10120122 17.10859 3.530236e-05
BICF2G63099370 0.5398855 0.13214522 16.69169 4.397309e-05
TIGRP2P325071  0.6802676 0.16971534 16.06635 6.116108e-05
                     Pc1df effAB effBB chi2.2df P2df
BICF2P1132186  1.476782e-07    NA    NA        0   NA
BICF2P942043   7.137105e-06    NA    NA        0   NA
BICF2P189945   7.280700e-06    NA    NA        0   NA
BICF2P217273   7.597807e-06    NA    NA        0   NA
BICF2P1496251  8.676625e-06    NA    NA        0   NA
BICF2S23629051 2.383242e-05    NA    NA        0   NA
BICF2S23621851 2.930795e-05    NA    NA        0   NA
BICF2P1330344  3.530236e-05    NA    NA        0   NA
BICF2G63099370 4.397309e-05    NA    NA        0   NA
TIGRP2P325071  6.116108e-05    NA    NA        0   NA
```

The `polygenic()` function may not work best for binary traits. If it cannot converge properly like below:

```
**********************************************
*** !!!BAD!!! convergence indicated by FGLS ***
**********************************************
```

you should try to change convergence parameters or to use its replacement `polygenic_hglm(GenABEL)`. However, you need devel version of GenABEL to do this. Visit http://genabel.r-forge.r-project.org/tutHowToInstallDevelVersion.html to learn how.

You may also want to try a combined mixed-model & SA approach. You can

do this by using strata information in the `mmscore()` function: `an.mmsa <- mmscore(h2a, data.clean, strata=pop, trait="binomial")`.

Each of the `*score` functions has the `times` parameter. By default it is set to 1, if you use a different number (N) – a number N of permutations will be performed to determine the experiment-wise association significances.

1.10 More Advanced Visualization

As you have probably noticed, all the `*score` functions are constructed in a similar way. You can play with them a lot, trying different setups, different sets of co-variates etc. However, in the end you always want to visualize your results in a nice way. Below, we will prepare:

- a custom Q-Q plot with p-values instead of χ statistics values and
- a custom Manhattan plot with nice colors and a threshold line.

Finally, we will learn how zoom on a specific region.
Some time ago we have written a custom function to make Q-Q plot looking nicer. You can define an appropriate function now:

```
> qqplot2 <- function(scan, type="P1df") {

            # Load the observed p-values
            lambda <- lambda(scan)$estimate
            lambdaSE <- lambda(scan)$se
            pvals <- scan[,type]
            logp = -log10(pvals)

            # Create uniform distribution (theoretical distr.)
            index <- seq(1, length(logp))
            uni <- index/length(logp)
            loguni <- -log10(uni)

            # Plot the Q-Q plot
            call <- as.character(scan@call)
            title <- paste("Q-Q plot for ", call[1],
            "(",call[2],") on ", call[3], sep="")
            subtitle <-paste("lambda = ", round(lambda,4),
            " (se = ", round(lambdaSE, 4),")",sep="")
            qqplot(loguni, logp, xlab="Theoretical",
            ylab="Observed", sub=subtitle, main=title, pch=19, cex=.5)
            abline(0,1, col="red")
    }
> # And  call the function
> qqplot2(an.sa, type="P1df")
```

Now, let us create a nice Manhattan plot (Figure 1.11):

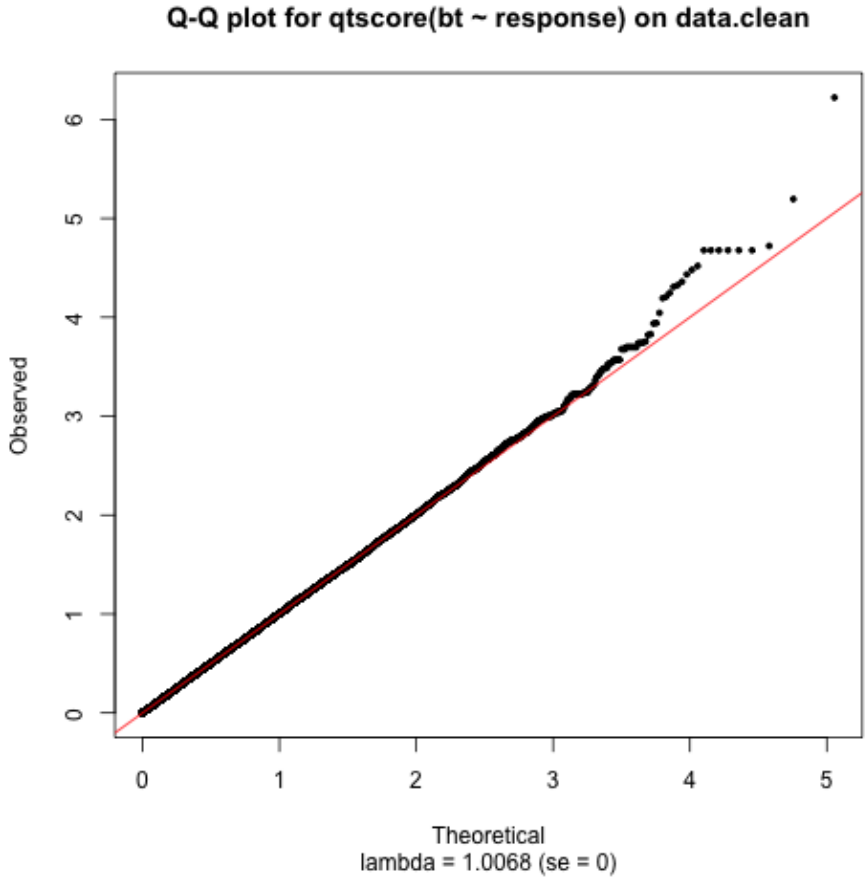

Figure 1.10: A custom Q-Q plot, the product of the `qqplot2()` function.

```
> an.gc <- qtscore(bt ~ response, data.clean)
> an.sa <- qtscore(bt ~ response, data.clean, strata=pop)
> # ^ Do association tests
>
> # Determine Bonferroni-corrected p-value threshold
> bonferroni <- -log10(0.05/nsnps(data.clean))
> # Plot the results
> plot(an.gc, col=c("red", "chocolate4"), pch=1, cex=.8, df="Pc1df")
> abline(h=bonferroni, col="red")
> add.plot(an.sa, col=c("lightblue", "navy"), pch=19, cex=0.4)
```

Finally, we promised to show you how to zoom in on a specific region. The result is presented in Figure 1.12

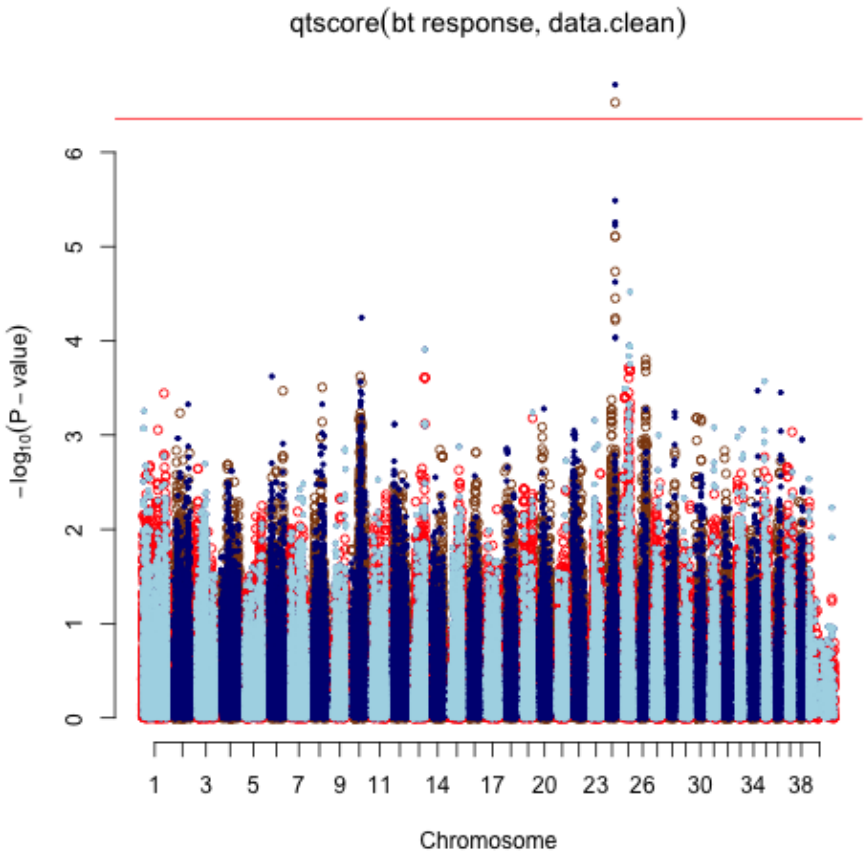

Figure 1.11: A Manhattan plot showing p-values determined using GC (red and chocolate) and SA (blue and navy blue). The red line represents Bonferroni-corrected p-value < 0.05 threshold.

```
> chr <- chromosome(gtdata(data.clean))
> chr2 <- snpnames(gtdata(data.clean))[chr == "2"]
> an.chr2 <- qtscore(bt~response, data.clean[,chr2], strata=pop)
> plot(an.chr2, pch=19, cex=.5)
```

If you want to zoom in even more, e.g. on the region between $2.0e8$ and $2.2e8$ on the same chromosome 2, you should re-fine the region:

```
> chr <- chromosome(gtdata(data.clean))
> chr2.region <- snpnames(gtdata(data.clean))[chr == "2" &
    map(gtdata(data.clean)) >= 2.0e8 &
    map(gtdata(data.clean)) <= 2.2e8]
```

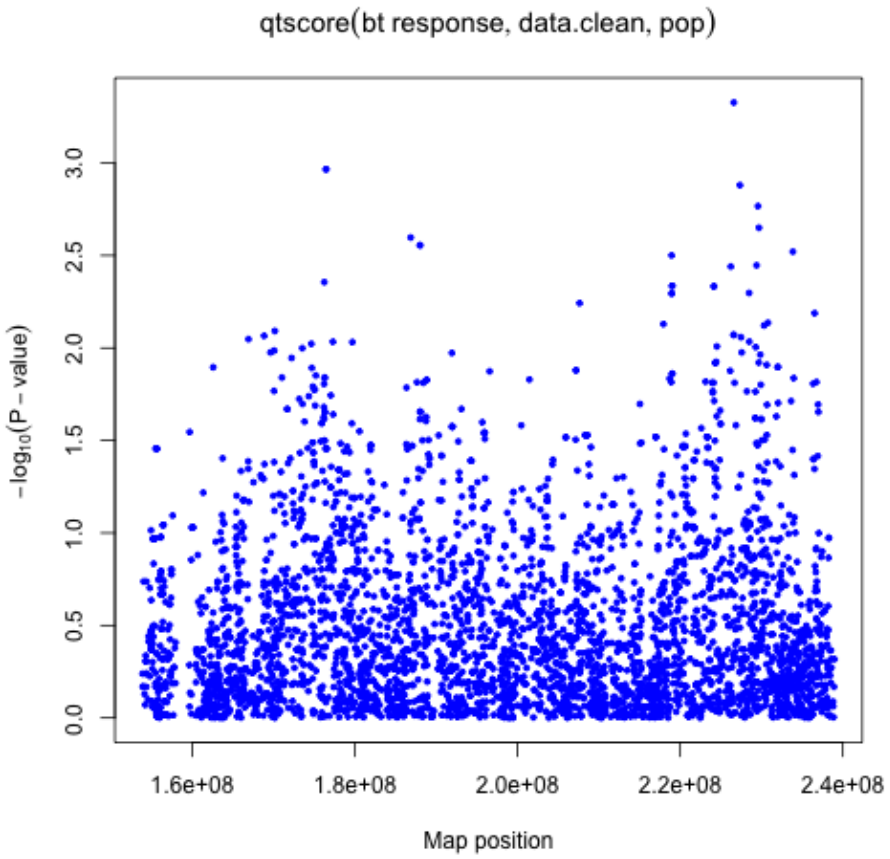

Figure 1.12: A Manhattan plot showing p-values of associations determined for chromosome 2 only.

* * *

Thank you! It was really nice working and sharing experience with you. We hope you enjoyed this tutorial.