Statistical analysis in genome-wide association studies

R and GenABEL-based tutorial

Marcin Kierczak*[1] and Mats Pettersson†[1]

[1]Computational Genetics Section, Department of Animal Breeding and Genetics, Swedish university of Agricultural Sciences, Uppsala, Sweden.

February 8, 2013

*marcin.kierczak@slu.se
†mats.pettersson@slu.se

1 Introduction

A genome-wide association study (GWAS) is a genetic association study performed on a genome-wide scale. The primary aim of a GWAS is to identify the genetic markers, typically single nucleotide polymorphisms (SNP), that are statistically significantly associated with the studied trait, the latter being either continuous such as weight or binary such as healthy/sick. During today's meeting we will use GenABEl, an R package specifically designed for carrying GWAS.

1.1 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.

1.2 Preparing work environment and loading the 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:

```
> # Load GenABEL
> library(GenABEL)
> # Load data
> data <- load.gwaa.data(
          phenofile="http://computationalgenetics.se/datasets/phenotype.dat",
          genofile="http://computationalgenetics.se/datasets/genotype.raw")
>
```

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.

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:

```
> # See all the phenotypic data (we do not show output here)
> phdata(data)


> # See phenotypes of the second individual
> phdata(data)[2,]
           id sex bt       ct
dog225 dog225   1  0 1.925575
       group antibodyLevel
dog225     3      1.569402
> # See phenotypes of the individuals 3 through 7
> phdata(data)[3:7,]
           id sex bt       ct
dog226 dog226   0  1 2.431597
dog227 dog227   1  0 4.175280
dog228 dog228   0  0 4.026864
dog229 dog229   0  1 2.972239
dog230 dog230   1  0 2.017118
       group antibodyLevel
dog226     3     0.7754572
dog227     1     1.3674980
dog228    NA     3.0488350
dog229     1     1.2468811
dog230     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 animals
> phdata(data)[1:5, c("sex", "bt")]
       sex bt
dog224   1  0
dog225   1  0
dog226   0  1
dog227   1  0
dog228   0  0
```

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])
  BICF2P1173580
            "1"
BICF2G630707846
            "1"
  BICF2P1383091
            "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 individuals?
> 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:

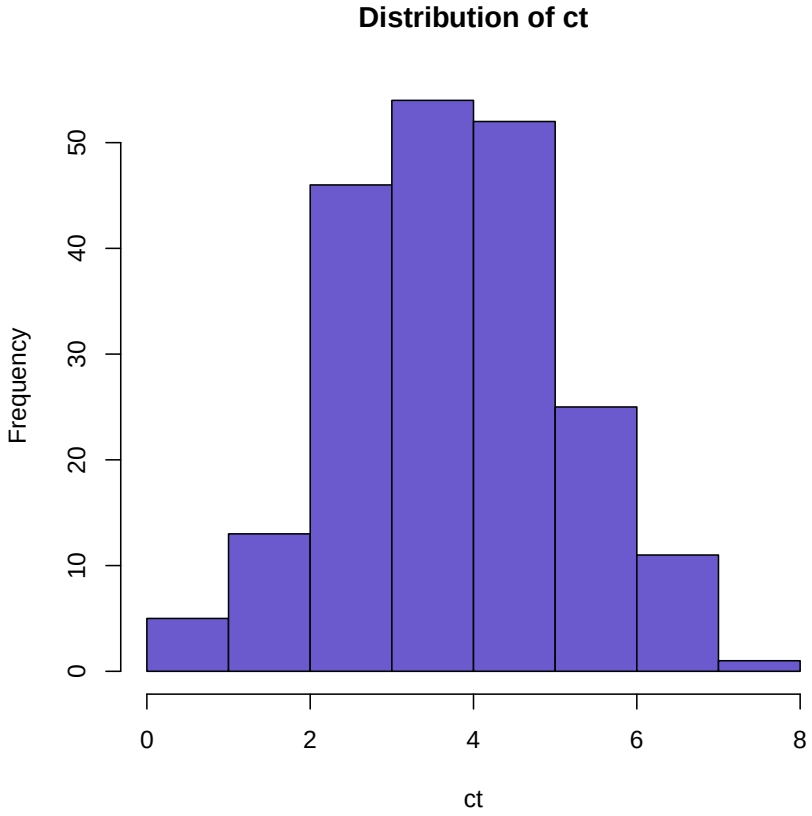
```
> # We do not show output in this tutorial!
> summary(gtdata(data))[1:5,]
```

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:

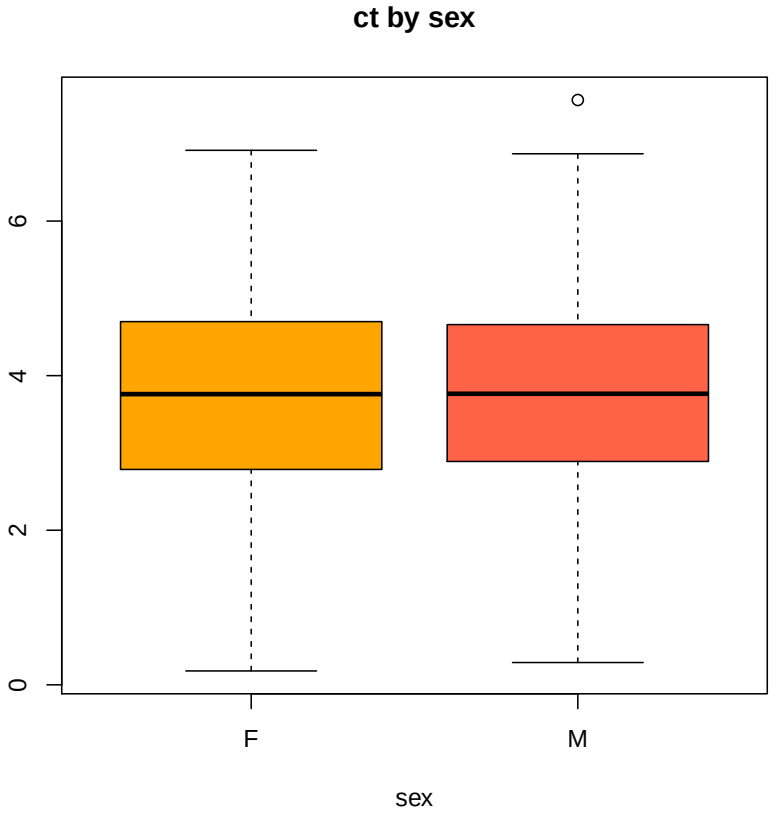
```
> # Get summary of all phenotypes
> summary(phdata(data))
      id
 Length:207
 Class :character
 Mode  :character




      sex
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
```

```
 Mean   :0.4589
 3rd Qu.:1.0000
 Max.   :1.0000

       bt
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.3073
 3rd Qu.:1.0000
 Max.   :1.0000
 NA's   :2
       ct
 Min.   :0.1797
 1st Qu.:2.8468
 Median :3.7654
 Mean   :3.7631
 3rd Qu.:4.6989
 Max.   :7.5664

     group
 Min.   :1.000
 1st Qu.:1.000
 Median :2.000
 Mean   :1.865
 3rd Qu.:3.000
 Max.   :3.000
 NA's   :66
 antibodyLevel
 Min.   :-0.4325
 1st Qu.: 0.9996
 Median : 1.4090
 Mean   : 1.4859
 3rd Qu.: 1.9477
 Max.   : 3.3858
 NA's   :1
> # Get summary for the continuous trait 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
> # Plot it as both histogram
> ct <- phdata(data)$ct
> hist(ct, col="slateblue", main="Distribution of ct")
```




```
> # And see whether continuous trait
> # depends on sex using boxplot
> boxplot(ct ~ phdata(data)$sex, col=c("orange", "tomato"),
                xlab="sex", names=c("F", "M"), main="ct by sex")
```



3 Preliminary quality control

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 $MAF < 10^{?8}$, i.e. only monomorphic ones 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.
Let's do quality control:

```
> qc <- check.marker(data, call = 0.95, perid.call = 0.95,
                                        maf = 1e-08, p.lev = 1e-08)
```

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

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

4 Testing for associations

Now, it is perhaps time to do a genome-wide association (GWAS) study. In our setup, we will analyze continuous trait and try to find markers that are associated with values of "ct". To accomplish this, we will test marker by marker using a statistical test (Cochran-Armitage trend test). This is implemented in GenABEL as `qtscore()` function.

```
> an <- qtscore(ct~1, data=data.qc, trait="gaussian")
```

Now the results of all the tests are stored in "an" object. We can plot per-marker p-values – the so-called Manhattan plot.

```
> plot(an, col=c("olivedrab","slateblue"), cex=.5,
          main="Manhattan plot")
```

Congratulations! It looks like we have an association on chromosome 2! But... wait. Can we really start celebrating? Let's examine genomic inflation factor, λ. If it is greater than 1, it means we have more low p-values than expected. This can be due to so-called confounding, e.g. cryptic relatedness or population structure. Let's examine λ and see how our p-values are distributed compared to the theoretical χ^2 distribution. We can use a Q-Q plot to accomplish this:

```
> estlambda(an[,"P1df"], plot=T)
$estimate
[1] 1.182951

$se
[1] 0.0008107524
```

Value of the genomic inflation factor $\lambda > 1$ indicates that we got some inflation in our results and we need to take care of this inflation. The simplest, but often efficient solution is called *g*enomic control. Genomic control is done by dividing each p-value by λ. This often restores expected distribution and removes the effect of inflation. GenABEL `qtscore()` performs genomic control by default and stores the result in "Pc1df" (P-value corrected 1 degree of freedom) column. Let us see if genomic control will help:

```
> plot(an, col=c("olivedrab","slateblue"), cex=.5,
          main="Manhattan plot", df="Pc1df")
```

```
> estlambda(an[,"Pc1df"], plot=T)
$estimate
[1] 1.000004

$se
[1] 0.0006853675
```

As you can see, now λ is really close to 1 and therefore we can assume that we have corrected inflation. Great! There is only one thing left now: as mentioned, our GWAS study consists of a large number of statistical tests – one per marker. There is always a possibility that by chance some of the p-values will reach signficance. The more tests we per form, the greater chance of such false positive. A simple way to correct for this is called *B*onferroni correction. Let us say that we set our p-value threshold to the standard 0.05, meaning we would like to keep the possibility of having false positive below 5%. Now, this is valid for one test. But here, we run as many tests as there are markers. To keep our threshold at this 5%, we need to divide each p-value by the number of tests. And, as on the Manhattan plot we are showing $-log_10(p-value)$, Bonferroni correction will take the following form:

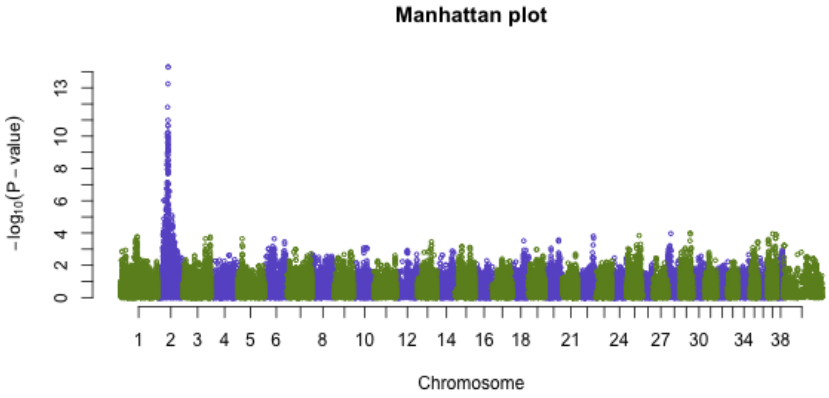


Figure 1: Manhattan plot showing the result of GWAS for continuous trait.

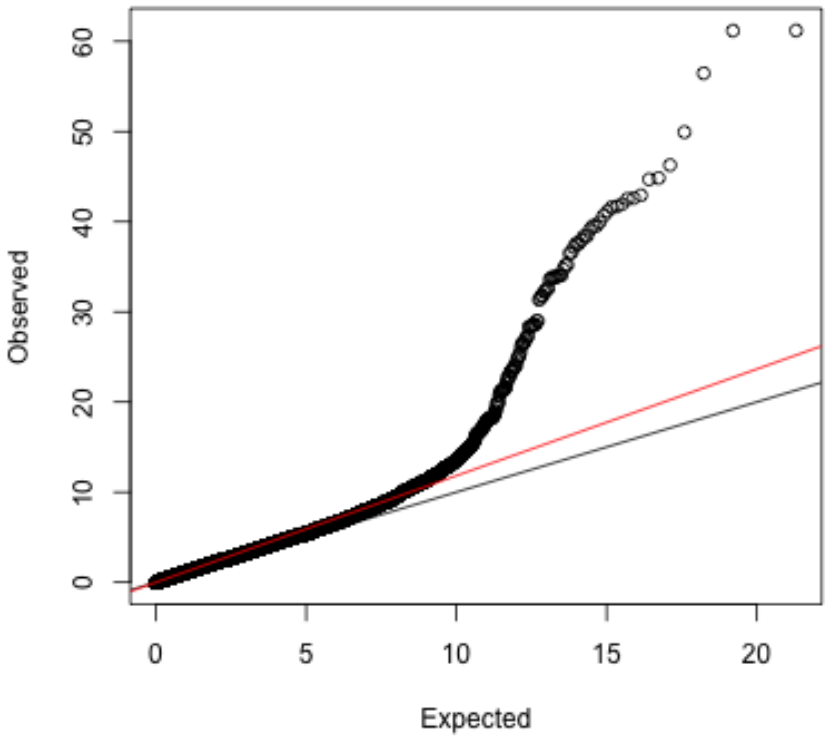


Figure 2: A Q-Q plot of $-log_{10}(p-values)$ obtained in GWAS for continuous trait.

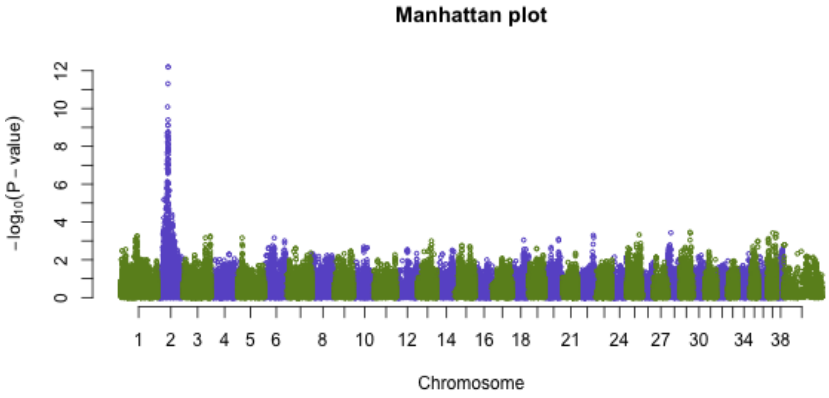


Figure 3: Manhattan plot showing the result of GWAS for continuous trait after genomic control.

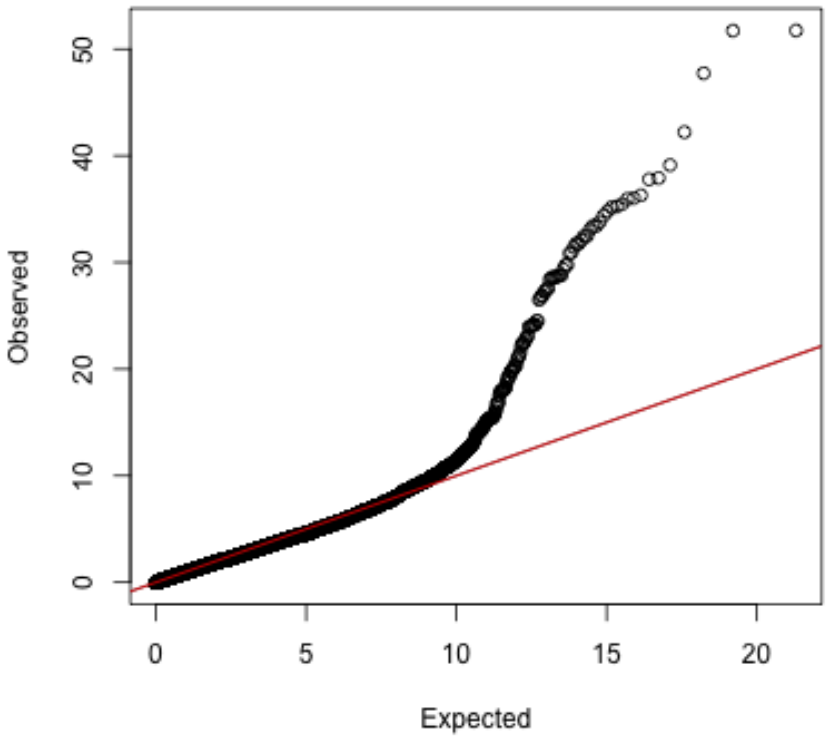


Figure 4: A Q-Q plot of $-log_{10}(p-values)$ obtained in GWAS for continuous trait after genomic control.

```
> pval.threshold <- 0.05
> bonferroni <- -log10(pval.threshold / nids(data.qc))
> bonferroni
[1] 3.608526
```

We can add Bonferroni threshold as a red line to our Manhattan plot:

```
> plot(an, col=c("olivedrab","slateblue"), cex=.5,
          main="Manhattan plot", df="Pc1df")
> abline(h=bonferroni, lty=3, color="red")
```

Bonferroni correction, however, is rather conservative and arbitrary. To actually get the right estimate of the true significance level, one has to run permutation tests. Permutation test is a test where we randomly shuffle phenotypic values and perform same association test as in for the original data. We do this several (typically several thousand) times and record the "best", i.e. the lowest p-value from each run. The position of our original best p-value on the distribution of the best p-values obtained with permutation test allows us assess the true chance of committing type-I error. Permutation test is very easy to perform in GenABEL, but can take significant amount of time. Therefore we just show how to call it:

```
> # Perform 10 000 permutation tests
> an <- qtscore(ct~1, data=data.qc, trait="gaussian", times=10000)
```

THANK YOU!

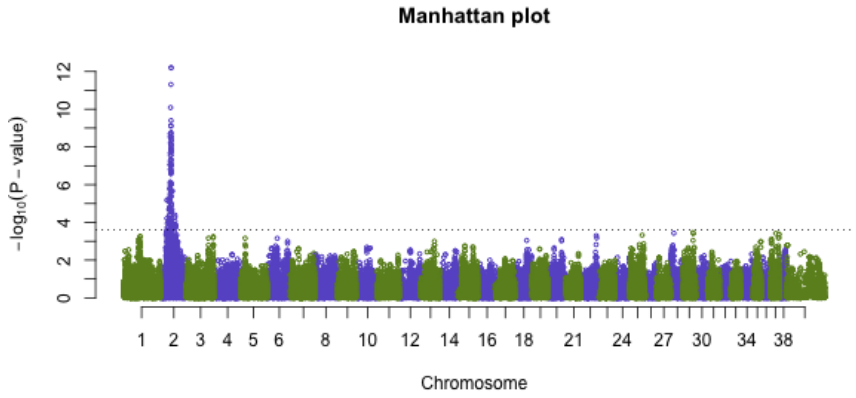


Figure 5: Manhattan plot showing the result of GWAS for continuous trait after genomic control. Bonferroni-corrected p-value threshold = 0.05 shown as red dashed line.