Dr. Shirin Glander

Statistical Genetics: Analyses with R

Contents

List of Tables

List of Figures

Preface

Hi there, this is my great book.

Why read this book

It is very important...

Structure of the book

Chapters 1 introduces a new topic, and ...

Acknowledgments

A lot of people helped me when I was writing the book.

Shirin Glander

About the Author

I am a bioinformatician at the University Hospital of Münster in Germany, where I'm responsible for Next Generation Sequencing analysis. I've earned my PhD from the University of Münster, where I worked on the link between flowering time and immune defense in plants using quantitative genetics and RNA-sequencing. My overarching interest has been evolutionary biology and genetics. At the moment, I am mainly working on questions relating to immunology - specifically inflammation, immune tolerance and auto-inflammatory diseases.

I'm a big fan of R and I write a blog[1] where I explore different data sets and techniques in R. I also teach ballroom and Latin dance courses.

You can find me and my package for gene expression analysis (exprAnalysis) on Github[2].

[1]https://shiring.github.io/
[2]https://github.com/ShirinG

The Basics of R

"R is an integrated suite of software facilities for data manipulation, calculation and graphical display." An Introduction to R - Notes on R: A Programming Environment for Data Analysis and Graphics version 3.4.0 (2017-04-21) by W. N. Venables, D. M. Smith and the R Core Team[3]

R has originally been developed for statistical analysis and is a powerful toolbox for a wide range of tasks pertaining to data handling, statistical modeling and visualisation. It consists of a core distribution that contains basic functions and additional optional packages for specific tasks.

R functions generally work with objects that have been assigned a value, e.g. a character string (`object = c("this", "is", "a", "string")`), numeric vector (`object = c(1, 2, 5, 70)`), model fit (`object = lm(speed ~ dist, data = cars)`), etc. Objects can be evaluated and compared.

This books assumes a basic knowledge about getting data into R, calling functions, etc. but it is beyond the scope of this book to provide an in depth introduction to R. Luckily, there are a large number of free online resources available, like

- An Introduction to R - Notes on R: A Programming Environment for Data Analysis and Graphics version 3.4.0 (2017-04-21) by W. N. Venables, D. M. Smith and the R Core Team[4]
- Data Camp[5]

[3]https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
[4]https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
[5]https://www.datacamp.com/courses/free-introduction-to-r

- Penn State Statistics resources[6]
- by Germán Rodríguez from Princeton University[7]
- etc.

Software information and conventions

Package names are in bold text (e.g., **rmarkdown**), and inline code and filenames are formatted in a typewriter font (e.g., `plot()`). Function names are followed by parentheses (e.g., `bookdown::render_book()`).

Some packages are available via CRAN[8], while others are hosted at Bioconductor[9]. I will provide package installation instructions at the beginning of each section, indicating where each package can be found. I will also be using the `library()` function, rather than `require()` for loading packages to make sure that we will get an error message in case packages have not been installed correctly.

The example workflows included are meant to illustrate the theoretical concepts and get you started on your own analysis. They are minimal examples of the necessary steps but are not meant to substitute the package manuals. When you want to apply the workflows to your own data, I highly recommend going back to the package documentation to find out about additional functions and using the `help()` function to explore parameter options. I will be using the same naming and code schemes as in the package manuals to make finding the relevant parts easy.

I used the **knitr** package and the **bookdown** package to compile this book. My R session information is shown below:

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
```

[6]https://onlinecourses.science.psu.edu/statprogram/r
[7]http://data.princeton.edu/R/introducingR.pdf
[8]https://cran.r-project.org/
[9]https://www.bioconductor.org/

```
## Running under: macOS Sierra 10.12.3
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.4.0  backports_1.1.0 bookdown_0.4    magrittr_1.5
##  [5] rprojroot_1.2   tools_3.4.0     htmltools_0.3.6 rstudioapi_0.6
##  [9] yaml_2.1.14     Rcpp_0.12.11    stringi_1.1.5   rmarkdown_1.5
## [13] knitr_1.16      stringr_1.2.0   digest_0.6.12   evaluate_0.10
```

1

Introduction to Statistical Genetics

2 Evolutionary Genetics

2.1 Evolution of genetic systems

2.2 Phylogenetics

2.3 Game Theory

2.4 Genetic Algorithms

2.5 Evolutionary Algorithms

3 Population Genetics

3.1 Hardy-Weinberg

3.2 Linkage Disequilibrium

4

Quantitative Genetics

4.1 QTL analysis

Quantitative Trait Loci (QTL) are regions in the genome that are associated with variation in a quantitative trait. Quantitative traits are phentoypes that can be measured on a continuous scale, like height, weight, etc.

QTL analysis (or QTL mapping) is typically done on experimental populations to find genes which contribute to the heritability of traits. Phenotype and genetic marker data are collected from every individual in the population. The general concept of QTL mapping is that we can then calculate the correlation between genotypes and phenotypes at each marker position and test whether they show a statistically significant association.

Let's consider the famous example of Doebley and Stec (1991), who assessed the variation of traits that discriminate commercial maize from its native relative teosinte. Teosinte is much smaller than maize as we know it today and one teosinte plant produces many ears, each of which has only two rows of seeds. But even though maize and teosinte look so completely different, they are still able to produce viable offspring together. Doebley and Stec (1991) utilized this and crossed the two plant species to produce an F1 generation, which were in turn self-pollinated. The resulting F2 population of maize-teosinte-hybrids showed a wide range of intermediate parental morphologies. Each of the F2 offspring was then genotyped at 58 locations in the genome, so that the quantitative trait information on morphology could be correlated with the genetic map. This analysis revealed that most of the morphological variation between maize and teosinte were the result of changes in only a handful of genes, one of which is the *tb1* (*teosinte branched 1*) gene.

4.1.1 Recombinant Inbred Lines (RILs)

RILs are experimental sister populations that have been produced by a very specific back-crossing scheme. The process is similar to Doebley and Stec's crossing of maize and teosinte: two homozygous parents are crossed to produce an F1 generation. Following the laws of genetics, each offspring's genome consists of a random combination of parental alleles and crossover (or recombination) events. Depending on the design, F1 offspring are usually either selfed or mated with a sibling to introduce another level of genetic recombination. The final generation is then inbred for many generations to obtain a collection of homozygous sister lines, each with a unique mosaic genome of parental alleles (Pollard, 2012).

4.1.2 QTL analysis in R

4.1.2.1 The qtl package

The most established R package for QTL mapping is Karl Broman's **qtl** package[1] (Broman et al., 2017). It implements several techniques for finding QTLs, like Hidden Markov Models (HMM), interval mapping, Haley-Knott regression and multiple imputation. It is very well documented and comes with extensive example data and code.

Here, I will introduce you to a basic QTL mapping workflow using the examples given in the package documentation and refer you to more complex analysis options where applicable.

4.1.2.1.1 Installation and loading the package

If this is the first time you are using the **qtl** package, you need to install it from CRAN. The following line of code checks whether you already have the package, and if not installs it.

```
pkg = "qtl"
if (system.file(package = pkg) == '') install.packages(pkg)
```

[1] http://www.rqtl.org/

You can then load the package:

```
library(qtl)
```

4.1.2.1.2 *Loading the data*

I will be using the example data on murine hypertension that is provided in the package (Sugiyama et al., 2001). In this dataset, we find information on 250 male mice from a reciprocal backcross between two strains: 1) the salt-sensitive C57BL/6J strain or 2) the inbred normotensive A/J strain. These mice were then given 1% salt water over two weeks and their hypertension blood pressure levels were measured.

```
data(hyper)
```

It is - as always - a good idea to familiarize yourself with the data to identify potential problems or errors before spending hours or days on an analysis that leads nowhere. The `summary()` function shows you the main properties of the data, like number of individuals and phenotypes, genotype information and proportion of missing data.

```
summary(hyper)
```

```
##     Backcross
##
##     No. individuals:    250
##
##     No. phenotypes:     2
##     Percent phenotyped: 100 100
##
##     No. chromosomes:    20
##         Autosomes:      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
##         X chr:          X
##
##     Total markers:      174
##     No. markers:        22 8 6 20 14 11 7 6 5 5 14 5 5 5 11 6 12 4 4 4
##     Percent genotyped:  47.7
##     Genotypes (%):
```

```
##           Autosomes:    BB:50.1  BA:49.9
##        X chromosome:    BY:53.0  AY:47.0
```

The `plot()` function produces plots that show missing genotypes, marker positions and the distribution of phenotypes or traits. Figure 4.1 will give you a first idea about the quality of your data.

The package manual[2] includes a description of various additional plotting functions, which I won't cover here.

```
plot(hyper)
```

4.1.2.1.3 *Genetic map estimation*

Before we proceed with the analysis, I typically recommend replacing the existing genetic map with an estimated one to reduce the potential errors. The genetic map represents all markers on a chromosome in a linear fashion. The 'est.map() function applies a Hidden Markov Model (Lander and Green, 1987) to estimate the map with an assumed genotyping error rate (*error.prob*).

Here, we can also specify the mapping function (*map.function*) that we want to use to convert genetic distance to recombination fraction. The distance between two markers is usually given as a unit of genetic linkage, called *centimorgan (cM)* One cM represents the distance with an average of 0.01 crossover events in one generation (i.e. 1% recombination). However, this representation of distance underestimates the actual recombination fraction, which is inherently not additive. With increasing distance the chance of double crossovers increases, so that they are in a way "invisible" to the traditional estimation of recombination distance.

Another reason why genetic maps based on recombination are biased is crossover interference, which described the phenomenon that a crossover event reduces the likelihood of another recombination event occur close by.

[2] http://www.rqtl.org/tutorials/rqtltour.pdf

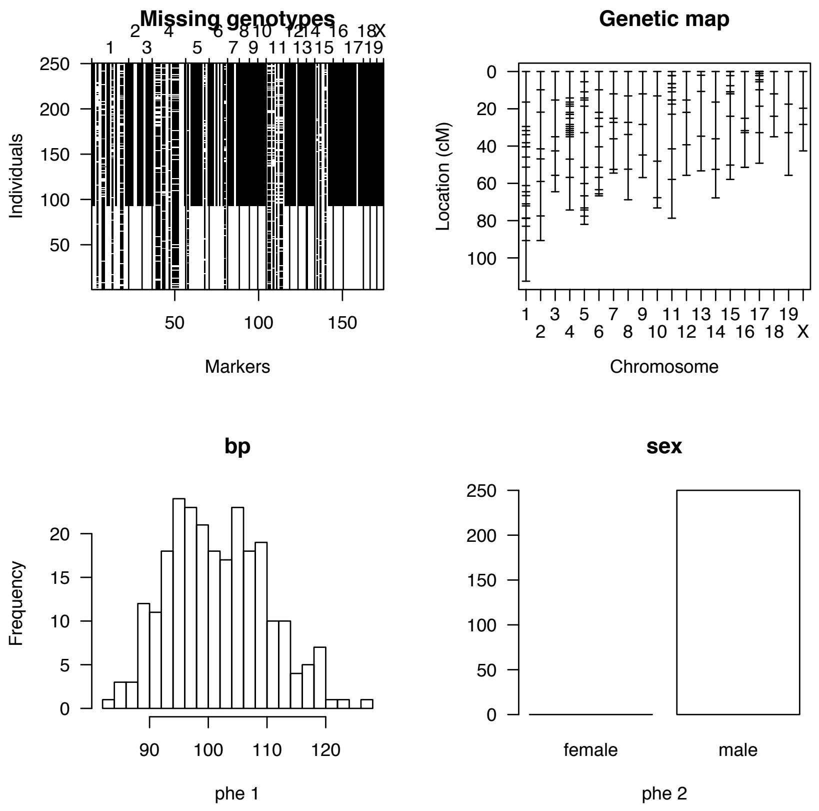

FIGURE 4.1
Plots providing an overview over missing genotypes, marker positions and the distribution of phenotypes or traits. Plots were generated with ‘qtl::plot()‘. In the upper left plot markers are shown against individuals, with black areas showing missing information. As we can see here, there is quite a bit of missing data. The upper right plot shows the marker positions on the chromosomes. Here, we can easily see that chromosomes 1, 4, 11 and 17 have clusters of more densely positioned markers. The lower plots show phenotype information, either as a histogram for continuous values or as a bar chart for categorical factors.

To correct for such biases, we can choose from the following mapping functions:

- **Haldane's** is the simplest mapping function and assumes a Poisson distribution for crossover events and does not consider interference.
- **Kosambi's** mapping function also considers interference and double crossovers but it can not calculate joint recombination probabilities for more than three loci.
- **Carter-Falconer's** mapping function can be extended to more complex interference rates.
- **Morgan's** mapping function assumes complete interference.

The two most widely used mapping functions are Haldane's (the default) and Kosambi's. For this example, using Haldane's should be sufficient. Because our example is a backcross, we can assume no interference, meaning that all crossovers are independent (Lynch and Walsh, 1998).

```
newmap <- est.map(hyper, error.prob = 0.0001,
                  map.function = "haldane")
hyper <- replace.map(hyper, newmap)
```

We can now estimate the recombination fractions between all pairs of markers. The `est.rf()` function also calculates the LOD scores. LOD stands for "likelihood of the odds" and is a measure of linkage. In QTL mapping we calculate LOD scores for the genetic markers and a threshold, above which we consider a QTL statistically significant in its association with the trait.

```
hyper <- est.rf(hyper)
```

The `calc.errorlod()` function calculates the genotyping errors according to Lincoln and Lander (1992). We can see which markers have an error LOD above a certain threshold (cutoff) with the `top.errorlod()` function (Table 4.1).

```
hyper <- calc.errorlod(hyper, error.prob = 0.0001)
te <- top.errorlod(hyper, cutoff = 3)
```

These potentially erroneous markers can then be removed or set to `NA`.

TABLE 4.1
Markers with an error LOD bigger than 3 under an error probability of 0.0001 indicate potential genotyping errors.

chr	id	marker	errorlod
4	102	D4Mit288	3.324582
4	107	D4Mit111	3.262205
4	216	D4Mit214	3.261092
11	57	D11Mit82	3.021105
11	118	D11Mit82	3.021105

```r
hyper.clean <- hyper
for (i in 1:nrow(te)) {
  chr <- te$chr[i]
  id <- te$id[i]
  mar <- te$marker[i]
  hyper.clean$geno[[chr]]$data[hyper$pheno$id == id, mar] <- NA
}
```

4.1.2.1.4 *Finding QTLs*

Now, we can proceed with the central step: mapping the QTLs.

Because the individuals in QTL studies are genotyped at specific marker locations throughout the genome, we inherently have to deal with the missing information about genotypes between markers. Hidden Markov Models (HMM) can help us overcome this problem by calculating genotype probabilities between markers based on the joint genotype distribution.

We first need to calculate these genotype probabilities using the `calc.genoprob()` function. We can define several parameters, like step size, the amount of error we want to allow for, the mapping function and step width. Here, we want to calculate genotype probabilities for every cM (step = 1), with a fixed step width and an error probability of 0.0001. As above, we are again using Haldane's mapping function.

```
hyper.clean <- calc.genoprob(hyper.clean, step = 1,
                             error.prob = 0.0001,
                             map.function = "haldane",
                             stepwidth = "fixed")
```

The simplest QTL model, we can run is single-QTL marker regression or interval mapping using the `scanone()` function.

These simple methods can give a good estimation of QTLs but they can also introduce bias, especially with multiple QTL in close proximity. More advanced mapping approaches, like Composite Interval Mapping (CIM) are discussed later on.

The first parameter we want to specify is the phenotype(s) and model (e.g. parametric or non-parametric) for mapping. Here, we want to use the first phenotype, i.e. the first column in our phenotype matrix, which follows a normal distribution. We can see the phenotype matrix by calling:

```
hyper.clean$pheno
```

Then, we need to specify the mapping algorithm we want to use. We can choose from several options. Here, I will only present the practical implications for each method. For a full discussion of the mathematical principles, see Lynch and Walsh (1998).

- **Marker regression**: Marker regression is by far the simplest approach to QTL mapping. Here, we calculate the association between phenotype and genotype at each marker position independently.

Because it is so simple, marker regression is seldom recommended to use. With interval mapping, a phenotype ~ genotype association analysis is performed for each flanking marker pair independently. This improves the approximation and gives confidence regions around QTL.

- **EM (Expectation-Maximization) algorithm**: EM is usually applied to maximum likelihood (ML) analyses of mixed models. It is an iterative process of calculating conditional probabilities and updating the ML estimates. This process is repeated until the estimates converge (Lan-

der and Botstein, 1989). If we have a reasonably dense marker map, the EM algorithm will converge on the global maximum.
- **(Extended) Haley-Knott regression**: Haley-Knott regression uses a simpler model than the EM algorithm (Haley and Knott, 1992). The extended Haley-Knott regression also considers variance is therefore gives improved approximations. Haley-Knott regression can give good approximations of the likelihood profiles for ML interval mapping but with more complex cases, it can be heavily biased.
- **Multiple imputation**: This method uses multiple rounds of imputing the unknown genotypes between markers and combines them into a final imputation model (Sen and Churchill, 2001). This allows us to perform a simple analysis of variance at each position in the genome. Multiple imputation needs much more computational power than simpler methods and will usually not outperform them with single-QTL models (it is much more advantageous with more complex multi-QTL models, however).

Here, I will show QTL mapping examples for the EM algorithm and for multiple imputation:

```
# EM algorithm
out.em <- scanone(hyper.clean, pheno.col = 1, model = "normal",
                  method = "em")
```

The multiple imputation method requires the use of the `sim.geno()` function before we call the mapping function. It calculates the joint genotype distribution based on the available marker information and uses it to perform the imputation of missin genotypes. With the `n.draws` parameter, we define how many imputations will be run. The more imputations steps we run, the more precise the genotypes but with increasing cost of computational time and power.

```
hyper.clean <- sim.geno(hyper.clean, step = 1, n.draws = 100,
                        error.prob = 0.0001,
                        map.function = "haldane",
                        stepwidth = "fixed")

out.imp <- scanone(hyper.clean, pheno.col = 1, model = "normal",
                   method = "imp")
```

Now that we have a LOD score for each marker position, we want to know which positions are significantly associated with the phenotype. To determine this, we will use the **scanone()** function again, but this time we want to calculate the genome-wide LOD score threshold with a permutation test. Above this threshold we can consider a QTL to be statistically significant. Here, I am using a similar call as before, but I am specifying that we want to use 1000 permutations.

```
operm.imp <- scanone(hyper.clean, pheno.col = 1, model = "normal",
                     method = "imp", n.perm = 1000)
```

The `summary()` function will tell us our genome-wide LOD score threshold for a given significance value (here 0.05).

```
lod <- summary(operm.imp, alpha = 0.05)
lod
```

```
## LOD thresholds (1000 permutations)
##     lod
## 5% 2.34
```

And now we can also refine our QTL results by including the significance threshold. This will give us the LOD score and estimated p-values for each marker above the threshold, around which we can now assume to have found a QTL for our trait of interest.

```
summary(out.imp, perms = operm.imp, alpha = 0.05, pvalues = TRUE)
```

```
##           chr   pos  lod  pval
## c1.loc97    1 100.3 3.60 0.005
## D4Mit164    4  41.6 8.08 0.000
## c15.loc32  15  37.5 2.35 0.050
```

Now that we have our QTL, we can plot them with the `plot()` function. Here, I am plotting the results from both, the EM algorithm and the multiple imputation method. As expected, they do not differ much. The horizontal dotted line shows the genome-wide LOD threshold and our two significant QTL pop up nicely on chromosomes 1 and 4.

```
plot(out.em, col = "blue")
plot(out.imp, col = "red", add = TRUE)
abline(h = lod[1], lty = 2)
```

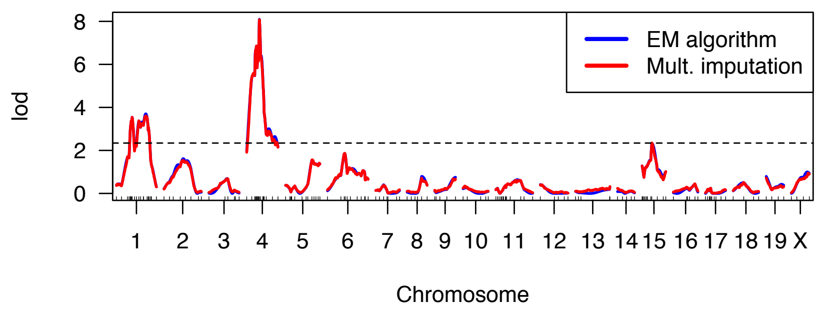

FIGURE 4.2
QTL plot showing LOD scores and QTL positions with EM algorithm and the multiple imputation. The dotted line indicates the genome-wide LOD score threshold for a $p < 0.05$. Every peak above this threshold is considered a QTL.

```
legend("topright", c("EM algorithm","Mult. imputation"),
       col = c("blue", "red"), lty = 1, lwd = 3)
```

4.1.2.1.5 *QTL interaction mapping*

4.1.2.1.6 *Covariates in QTL models*

4.1.2.2 QTL Analysis using Bayesian Interval Mapping (“qtlbim” package)

4.2 Gene x Environment interactions

4.3 Variance and heritability

Exercises: Quantitative Genetics

5 Genetics of complex diseases

5.1 Genome Wide Association Studies (GWAS)

5.2 Pedigree analysis

6

Genetic Epidemiology

6.1 Infection models

A

More to Say

Yeah! I have finished my book, but I have more to say about some topics. Let me explain them in this appendix.

To know more about **bookdown**, see https://bookdown.org.

Bibliography

Broman, K. W., Wu, H., with ideas from Gary Churchill, Sen, S., contributions from Danny Arends, Corty, R., Flutre, T., Jansen, R., Prins, P., Ronnegard, L., Shah, R., Shannon, L., Tran, Q., Wolen, A., and Yandell, B. (2017). *qtl: Tools for Analyzing QTL Experiments*. R package version 1.41-6.

Doebley, J. and Stec, A. (1991). Genetic analysis of the morphological differences between maize and teosinte. *Genetics*, 129(1):285–295.

Haley, C. S. and Knott, S. A. (1992). A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. *Heredity*, 69(4):315–324.

Lander, E. S. and Botstein, D. (1989). Mapping mendelian factors underlying quantitative traits using rflp linkage maps. *Genetics*, 121(1):185–199.

Lander, E. S. and Green, P. (1987). Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Sciences*, 84(8):2363–2367.

Lincoln, S. E. and Lander, E. S. (1992). Systematic detection of errors in genetic linkage data. *Genomics*, 14(3):604 – 610.

Lynch, M. and Walsh, B. (1998). *Genetics and Analysis of Quantitative Traits*. Sinauer.

Pollard, D. A. (2012). *Design and Construction of Recombinant Inbred Lines*, pages 31–39. Humana Press, Totowa, NJ.

Sen, Ś. and Churchill, G. A. (2001). A statistical framework for quantitative trait mapping. *Genetics*, 159(1):371–387.

Sugiyama, F., Churchill, G. A., Higgins, D. C., Johns, C., Makaritsis, K. P., Gavras, H., and Paigen, B. (2001). Concordance of murine quantitative trait loci for salt-induced hypertension with rat and human loci. *Genomics*, 71(1):70 – 77.

Index