Package noia: tutorial

Arnaud Le Rouzic

August 5, 2010

The package noia is an implementation for R of the Natural and Orthogonal InterAction model (NOIA), a statistical framework aiming at estimating and manipulating genetic effects of quantitative characters. This page is an informal tutorial describing the practical use of the software, as well as some basic concepts in quantitative genetics modeling. It comes as a complement of several sources of information:

- The statistical model is originally described in Álvarez-Castro & Carlborg (2007) Genetics 176:1151-1167.
- The application of the model to read data by least square regression is the topic of Álvarez-Castro, Le Rouzic & Carlborg (2008) PLoS Genet. 4:e1000062.
- The implementation of the model in a software is described in Le Rouzic & Álvarez-Castro (2008) Evol. Bioinform. 4 225-235.

The last version of the package can be downloaded on the official CRAN site (http://cran.r-project.org/web/packages/noia/index.html), where the reference manual is also available. More information about R in general, including manuals and tutorials, can be found on http://www.r-project.org.

For clarity, this tutorial is devoid of scientific citations: these can be found in the references above.

Contents

1 Context **2**

2 Models of genetic effects **2**

2.1 A simple example in haploid species 2

2.2 Genetic effects depend on the reference genotype 2

2.3 Average reference point 3

2.4 The diploid case 3

2.5 Terminology and nature of genetic effects 3

3 The NOIA model **4**

3.1 General framework 4

3.2 Orthogonality 5

3.3 Least square regression 5

3.4 The multilinear model 5

4 The `noia` package **6**

4.1 What the package can do 6

4.2 Data input 7

4.3 Linear regression 9

4.4 Genetic models 12

4.5 Genotype-phenotype map 14

4.6 Multilinear regression 14

5 Getting involved **16**

5.1 Questions and bug reports 16

5.2 Contributions 16

1 Context

The determination of the genetic basis of traits of interest remains an important activity in modern genetics. Yet, the process towards the understanding of the genetic architecture of quantitative characters does not stop with the identification of the DNA sequences that generate differences within an experimental population. Indeed, of tremendous interest is the quantification of the effect of each gene variant, by itself and through its interactions with other genes. This document provides a short and (hopefully) practical introduction to genetic effect models, presents its recent advances, and describes how to use this theory to analyze real datasets.

2 Models of genetic effects

Genetic effects are a way to quantify the effect of alleles or allele combinations in terms of phenotypic units. There are different ways to compute them, corresponding to different meanings of the obtained values.

2.1 A simple example in haploid species

The simplest case can be found in haploid organisms, where each individual carry only one allele at each gene. Consider a single locus and two alleles A and a. Individuals of genotype a weight in average 10g, individuals of genotype A weight 13g in average. The effect of substituting a by A is +3g (and conversely, the effect of substituting A by a is -3g). More formally, in the background defined by the genotype a, allele A has an effect of +3g: this is the average impact of substituting an allele A in a reference population of genotype a.

2.2 Genetic effects depend on the reference genotype

Cases in which only one gene is involved in the expression of a complex phenotype are, however, exceptional. Most of the time, quantitative traits are polygenic, and the effects of various genes do not necessarily add up. In practice, they actually never add up perfectly, at least because of measurement and sampling error, but mostly because biological processes are not additive. Such deviations from additivity (interactions) between genes are referred to as *epistasis*.

Back to the haploid example, but with two diallelic genes, say, a and A alleles at a first locus, and b / B alleles at a second locus. Four genotypes are possible, with for instance ab: 10g, Ab: 13g, aB: 10g, AB: 15g. This example aims at illustrating a key property of genetic effects: genetic effects depend on the reference point. If the reference genotype is ab, then the genetic effect at the first locus is +3g: substituting an allele A in genotype ab leads to genotype Ab, which is 3g heavier than ab. In this ab background, the

effect of the second locus is 0g (substituting a B allele does not change the phenotype). Adding both A and B alleles at the same time increases the phenotype by 5g, *i.e.* 2g more than if there were no interactions: the interaction effect is +2g.

On the contrary, if the genetic background is the genotype aB, substituting an allele A will lead to a gain of +5g. The effect of a substitution at the first locus is thus +3g in one genetic background, and +5g in another background.

The fact that genetic effects change depending on the reference point drives the whole complexity of the field, and has generated a number of scientific contributions (references can be found in the papers listed at the top of the page).

2.3 Average reference point

So far, the examples used specific genotypes as reference points. It is common to use genotype combinations as reference points instead, according to some arbitrary preference or to empirical considerations. For instance, the average effect of the allele A above would be +4g in a background constituted by 50% b and 50% B genotypes. Using such *population* reference points is important when calculating the actual genetic effects in a real population, since they directly condition the decomposition of genetic variance.

2.4 The diploid case

The presence of heterozygotes in diploid species makes the whole picture slightly more complicated, and genetic models turn out to be actually needed. With one locus and two alleles a and A, three genotypes are possible (aa, aA and AA). If the trait is perfectly additive, the heterozygote Aa is exactly between aa and AA. For instance, if aa: 10g, aA: 12g, AA: 14g, adding up one A allele increases the weight by 2g (and consequently, adding two A alleles increases the phenotype by 4g).

If the heterozygote is not exactly mid-way between both homozygotes, the discrepancy will be accounted for by a 'dominance' effect. For instance, if aa: 10g, aA: 13g, AA: 14g, the dominance effect will be +1g, since the heterozygote aA is 1g heavier than what would be expected under a strict additive case. The difference between AA and aa is still 4g, so that the additive effect for this locus is 2g, *i.e.* half the effect of substituting two alleles A.

Epistasis is treated in the same way as for the haploid case, and has the same consequences: genetic effect change depending on the background in which they are considered. Measuring epistasis-related genetic effects is significantly more complex, since three kinds of interactions can be defined when two loci are considered: the additive by additive (A×A) interaction describes how additive effects are affected by each other (*i.e.* how much the double homozygote $AABB$ differs from what would be expected under additivity), dominance-by-dominance (D×D) quantifies the discrepancy between the actual value of the double heterozygote $aAbB$ and the expectation if there were no epistatic interactions, and additive-by-dominance (A×D) effects measure the impact of epistasis on genotypes like $AAbB$ and $aABB$. In total, for two loci, nine genetic effects are necessary to fully describe the nine genotypic values: the reference effect, two additive effects, two dominance effects, one A×A effect, one D×D effect, and two A×D effects. For L loci, 3^L effects can be described, with interactions up to L levels (for instance, between 3 loci, it is possible to consider *e.g.* A×A×A or A×D×D effects, which are more and more difficult to interpret when the number of loci increase).

In the same was as for the haploid case, genetic effects can easily be defined from any reference point, including a mixture between genotypes.

2.5 Terminology and nature of genetic effects

The history of the definition of genetic effects is interesting by itself. The concepts on which quantitative genetics are founded were mainly described by R.A. Fisher, and they were mostly statistical: populations are characterized by the decomposition of their genetic variance components (additive variance, dominance variance, etc), but of little interest were the individual genetic effects. The theory was later extended to

account for variance components related to epistasis, but a lot of confusion remained about the nature of individual genetic effects, at least among non-specialists, until the 1990s. It is now admitted that there are several kinds of genetic effects, which describe different aspects of the genotype-phenotype relationship; here is a tentative classification distinguishing four cases:

1. When the reference genotype is a single genotype, genetic effects describe the phenotypic consequences of substituting alleles in this genotype.
2. When the reference genotype is an arbitrary, constant mixture of genotypes, genetic effects describe the substitution effect in such a background.
3. When the reference genotype is a mixture of genotypes defined in such a way that they match the population under study, genetic effects can still describe the effects of substituting alleles, but these will be average effects in a population and will depend not only on the genotype-phenotype map, but also on the allele and genotype frequencies in the population.
4. When genetic effects aim at providing a basis for the calculation of some statistical features of the population (*e.g.* variance decomposition), they are determined according to some rules (*e.g.* independence) which do not necessarily fit with the description of substitution effects.

There is no universally accepted terminology to describe these effect types. 'Physiological' effects (or, sometimes, 'compositional', 'genetical', 'biological'...) are associated to type 2, but they can probably fit with the spirit of type 1 effects. 'Functional' effects, most of the time used as a synonym of 'physiological', also frequently stands for type 3. Type 4 effects are generally called 'statistical'.

This complexity is also due to the fact that effect types can easily overlap. Type 3 effects measured in a monomorphic population exactly correspond to type 1 effects. Moreover, type 4 effects exactly overlap with type 3 effects in many situations (*e.g.* random mating populations), the difference between them being of the same nature as the additive vs. average excess effects defined by Fisher.

3 The NOIA model

The Natural and Orthogonal InterAction model is a statistical framework aiming at unifying, extending and simplifying existing models of genetic effects.

3.1 General framework

The formulation of the NOIA model relies on matrix algebra, and the notation here will follow (more or less) the initial publication. The model proposes to manipulate and compute genetic effects in diploid, diallelic populations, for any number of loci L. The two important vectors are $\mathbf{E}_m$, the vector of genetic effects in the framework of the model m (remember, the genetic effects depend on the model in which they are considered, including *e.g.* the background genotype), and $\mathbf{G}$, the vector of genotypic values, both of them being of length 3^L. Genotypic values are constant and do not depend on any model, they are a property of the genotype-phenotype map. For instance, for $L = 1$, $\mathbf{E}_m = (R_m, a_m, d_m)$ (R_m for the reference point, a_m for the additive effect, d_m for the dominance effect), and $\mathbf{G} = (G_{aa}, G_{Aa}, G_{AA})$ (the three genotypic values). $\mathbf{E}_m$ and $\mathbf{G}$ are linked by a $3^L \times 3^L$ design matrix, $\mathbf{S}_m$, which actually 'contains' the description of the model. Computing the genetic effects from the genotypic values in the context of model m is straightforward: $\mathbf{G} = \mathbf{S}_m\mathbf{E}_m$, and the opposite operation is $\mathbf{E}_m = \mathbf{S}_m^{-1}\mathbf{G}$. Combining the two previous equations provide the 'change of reference' operation, allowing to compute the genetic effects in the framework of a model m_2, given the genetic effects in a model m_1: $\mathbf{E}_{m_2} = \mathbf{S}_{m_2}^{-1}S_{m_1}\mathbf{E}_{m_1}$.

Details on how to build matrix $\mathbf{S}_m$ according to model m will not be detailed here, but an important information is that it is easy to compute $\mathbf{S}_m$ for any number of loci from the one-locus $\mathbf{S}_m$, assuming linkage equilibrium. This simple algebra operations make thus possible to build models of arbitrary complexity.

3.2 Orthogonality

Orthogonalization of genetic models have been an objective of many quantitative geneticists since almost a century. When an orthogonal model has to be used depends on the kind of inference one wants to draw from the data, but it is doubtless that such a model would be of general interest for the scientific community. An orthogonal decomposition of genetic effects have three major practical properties:

- It provides statistically independent (uncorrelated) genetic effects.
- It leads to a proper decomposition of genetic variance: the sum of variance components (var(A), var(D), etc.) is exactly equal to the explained genetic variance, which is an expected result in quantitative genetics.
- It makes model selection approaches much easier, since it is possible to remove effects without affecting the others.

The NOIA model provides a matrix $\mathbf{S}_{\mathrm{noia}}$ which always give orthogonal estimates $\mathbf{E}_{\mathrm{noia}}$ in the one-locus case, and multilocus orthogonal estimates if there is no linkage disequilibrium. Although this model is still in developemnt (so far, linkage disequilibrium still breaks independence), this constitutes a step forward towards an orthogonal decomposition of genetic effects

3.3 Least square regression

The equation $\mathbf{E}_m = \mathbf{S}_m^{-1}\mathbf{G}$ makes it possible to estimate genetic effects when genotypic values are known, but this is not very common nor convenient. When a small amount of loci are considered, it is possible (but not recommended) to calculate the average phenotypic value for each genotype, but this method will break rapidly with the number of loci, since some genotypic combinations will be missing or estimated with a large error due to a small sample size.

Instead, the NOIA framework proposes to estimate genetic effects by a least square regression, provided that a set of individuals have been genotyped and phenotyped for loci of interest. If $\mathbf{Y}$ is the vector of phenotypes and $\mathbf{Z}$ is a matrix reflecting the genotype of each individual (details are provided in the publications listed at the beginning of this tutorial), the vector of genetic effects can be estimated from the linear regression: $\mathbf{Y} = (\mathbf{Z}\mathbf{S}_m)\mathbf{E}_m + \mathbf{e}$, where $\mathbf{e}$ denotes the vector of residuals. Genotypic values, as well as genetic variances, can be easily calculated from $\mathbf{E}$.

3.4 The multilinear model

The multilinear model of genetic interactions is not part of the NOIA model as such. Defined by T.F. Hansen and G.P. Wagner in the early 1990s as a way to account for evolutionary-relevant genetic interactions, the multilinear model constitutes an alternative to the usual decomposition of genetic variance in additive-by-additive, additive-by-dominance, and dominance-by-dominance epistasis, which are easily defined statistically, but remain difficult to interpret in terms of evolutionary consequences. Accessorily, Hansen & Wagner's model is the first individual-reference functional model of genetic effects.

Back to the haploid example: two loci with two alleles each, leading to four haploid genotypes: ab, Ab, aB, and AB. The usual way to define genetic effects, setting arbitrarily the reference to ab, is to consider two additive genetic effects (for the first locus, a_1 is the phenotypic difference between Ab and ab, and for the second locus, a_2 is the difference between aB and ab). The phenotypic value of genotype AB is then the reference, plus the two additive effects, plus an independent (additive by additive) epistatic term: $AB = R + a_1 + a_2 + aa$ (note that aa is a new genetic effect, not a multiplied by a — the historical notation is somehow confusing).

The multilinear model is based on the assumption that genetic interactions scale with the additive effect of allelic substitutions. The previous example, taken into the framework of the multilinear model, would be $AB = R + a_1 + a_2 + a_1 \cdot a_2 \cdot \varepsilon$: the epistatic term scales with the product of additive effects, the scaling factor, ε, describing the strength of epistasis. If $\varepsilon = 0$, there is no epistasis, if $\varepsilon > 0$, epistasis is

"positive", *i.e.* the effect of substitutions amplify each other (the phenotypic value of a genotype combining two substitutions will be higher than the sum of the two substitutions taken independently), and if $\varepsilon < 0$, epistasis is "negative", *i.e.* the substitutions tend to cancel each other. As for all other genetic effects, the value of this parameter depends on the genotype (or the mixture of genotypes) chosen as a reference. Note that the concept of positive or negative epistasis strongly relies on the direction on which the trait is considered: positive epistasis for *e.g.* body darkness is the same as negative epistasis for lightness.

The most detailed version of the multilinear model involves a parameter ε_{ij} for each pair of loci i and j. This setting can be cumbersome when many loci are involved, and it is possible to replace many ε_{ij} by a composite directional epistasis parameter, ε_c. This composite parameter describes the 'average' directionality of the genotype-phenotype map, *i.e.* its general 'curvature', which is known to have a direct influence on the evolutionary properties of the population.

Extending the model for diploid populations, including dominance, is mathematically tractable, and such a model can handle an infinite number of loci and alleles per locus. The theory also extends to higher-order interactions, *e.g.* combining three substitutions at three different loci can be described by three two-order epsilons and one third-order epsilon. The complexity of the model thus increases with the number of loci. In the implementation described further, only diallelic models were considered, and the order of epistasis was restricted to pairwise interactions for the multilinear model.

4 The `noia` package

The package `noia` for `R` contains a set of functions implementing the NOIA model and facilitating the manipulation of genetic effects and genotype-phenotype maps. This tutorial will not attempt at providing a comprehensive documentation on all functions (see the pdf documentation instead), but rather to present a quick introduction to the main features of the program, and reproduce the steps that could be required for a real-data analysis.

The most universal way to install the package is through the `install.package()` function:

```
> install.package("noia")
```

This will automatically download the official `noia` package from the CRAN web site (`http://cran.r-project.org/`) and install it. Alternative ways to install packages may exist through the interface of R, different under each operating system.

Before use, it is necessary to load the package with:

```
> library("noia")
```

4.1 What the package can do

The package `noia` is primarily designed to analyze empirical data sets, but of course it is possible to run it on simulated data for a more theoretical approach. In any case, the program needs to be fed with a genotype-phenotype data set, including (i) a single phenotypic measurement for each individual, and (ii) some information about the genotype of each individual of the experimental population, for a limited set of loci known or suspected to be associated with the phenotype. Figure 1 summarizes the role which can be played by this package when dissecting the genetic architecture of a particular trait.

There are many things that the package cannot do:

- The `noia` package is not a QTL mapping program. The genetic information has to be provided for a limited, selected set of loci that are known to influence the phenotype. Typically, 'noia' can be used to analyze further the results of a QTL mapping experiment, once a set of major loci has been identified.
- So far, the package can analyze only single traits.

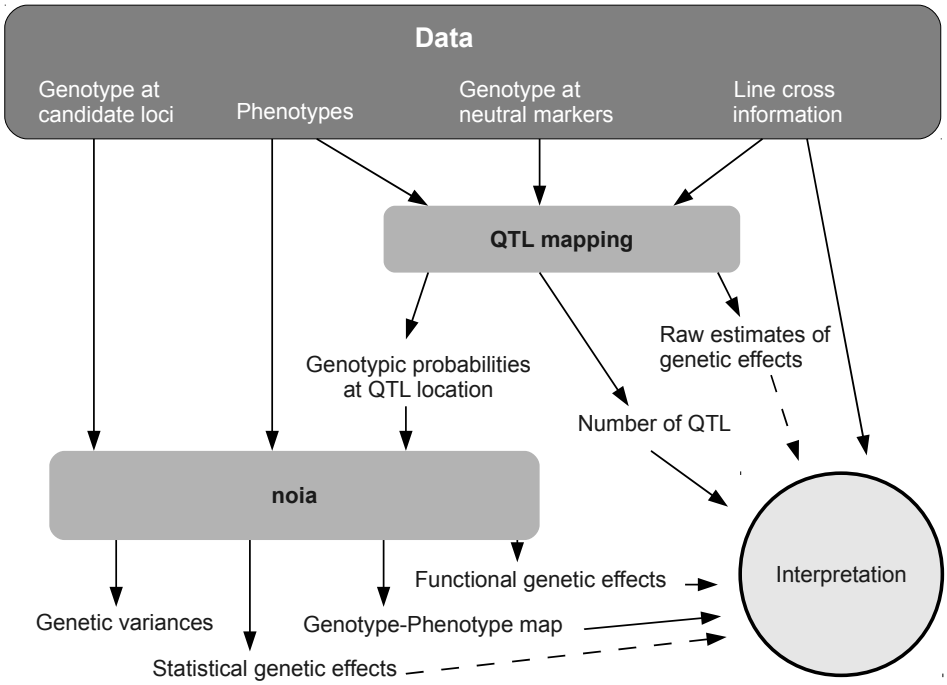

Figure 1: Diagram representing the use of the `noia` package in the process of dissecting genetic architectures. Genetic models implemented in the QTL mapping programs are in general not flexible nor precise enough to address all kinds of questions, and the results provided by this package contribute to improve the interpretation of raw QTL scans.

- So far, the package only considers diallelic loci. The theory for more than two alleles exists, but is not implemented yet.
- So far, the package does not implement indicence matrices that maintain strict orthogonality despite missing genotype information.
- Although the theory is designed to cope with any number of loci, the program will rapidly hit the memory and/or computational limits of a regular desktop computer above 12-15 loci, depending on the complexity of the models.
- Even if some effort has been made to clarify the documentation and make the functions easy to use, the package is unable to teach quantitative genetics, nor it can run a meaningful analysis by itself.
- The package is not designed to be an introduction to `R`. Frequently, some little formatting work on the data set or on results may be necessary, variables will need to be stored, some results might require further analysis through statistical functions that are not provided along with the package. Basic knowledge of the `R` syntax is therefore a prerequisite for a comfortable use of the 'noia' package.

4.2 Data input

The package needs two pieces of data: (i) phenotypes and (ii) genotypes for each of the n individuals of an experimental population. Phenotypes have to be stored as a vector of n numbers (which can be negative or null). Note that the program will not complain if you enter binary traits (e. g. 0 and 1) or meristic traits (*e.g.* number of eggs layed for 10 days), but just be sure of what you are doing: `noia` is built as a model for quantitative, continuous characters. Missing values (`NA`) are allowed, individuals with missing phenotypes are simply removed before the analysis (so don't spend too much time with them). As a simple example, let consider the following vector of phenotypes:

```
> phen <- c(10,12,9,7,11,16,10,11)
```

There are two ways to provide the genotypic data, one for perfect information, the `gen` method, (*e.g.* genotypes at marker positions), and one for incomplete information, the `genZ` method (*e.g.* QTL peak between two markers). These methods are alternative methods, *i.e.* they cannot be combined, but note that `genZ` can always bring the same information as `gen`, although the contrary is not necessarily true.

The `gen` method is the simplest. A `gen` data set is a matrix of L columns and n lines, where L is the number of loci and n is the number of individuals. The matrix can contain only three possible genotypes; 1 for the homozogote aa, 2 for the heterozygote aA, or 3 for the homozygote AA. Missing values (`NA`) are allowed. The following matrix is an example of a 1-locus data set:

```
> genot1 <- as.matrix(c(1,2,1,3,2,2,3,NA))
```

A multilocus data set would be similar, with more than one column to the matrix.

The `genZ` method requires three columns per locus. The first column corresponds to the probability of being of genotype aa, the second column of genotype aA, and the third column of genotype AA. If the genotype is perfectly well known, then this probability will be 1 for this genotype, and 0 for the others. For instance, the following `genZ` matrix is exactly identical to the previous `gen` example:

```
> genot2 <- matrix(c(
1,  0,  0,
0,  1,  0,
1,  0,  0,
0,  0,  1,
0,  1,  0,
0,  1,  0,
0,  0,  1,
2/7,3/7,2/7
),by.row=TRUE, ncol=3)
```

Note that the missing value was not coded as a probability of 0.333 of being of each genotype, but rather used the estimated genotypic frequencies from other loci. This is the default behavior in the package, because it does not affect genotypic frequencies. Since it is always possible to provide a `genZ` matrix instead of `gen`, it is easy to code the missing values otherwise if the default behavior is not acceptable.

`genZ` matrices for L loci will have 3^L columns, the three first ones being for locus 1, the three following for locus 2, etc.

A direct interest of `genZ` matrices is to feed `noia` with QTL genotypes when QTLs are located between markers. In general, QTL mapping programs estimate genotypic probabilities between markers (interval mapping), and can provide such estimates (most of the time calculated by a Haley-Knott regression, but several alternatives exist). When these data are provided by two numbers, generally called a and d (their meaning is totally unrelated to the additive and dominance effects), a simple way to convert them into probabilities for the `genZ` matrix is: $\mathrm{P}(aa) = (1 - a - d)/2$, $\mathrm{P}(aA) = d$, $\mathrm{P}(AA) = a + (1 - a - d)/2$. Note that the whole process is completely symmetric, if aa and AA are reversed, all computed effects will be of opposite sign, but the analysis will not be affected.

Obviously, it would be boring and time-consuming to type the data directly in `R`, and the easiest is to import a file, and extract the phenotypic and genotypic data from it. Let's consider the following file (text format, tabulations between columns):

```
phen    gen1    gen2    gen3
10.8    1       2       1
12.2    2       3       2
9.6     1       2       2
8.4     3       2       3
5.9     2       2       2
```

This file (named *file.txt* for the example) can be imported in R with the following command:

```
> data <- read.table("file.txt", header=TRUE)
```

The object data is a data frame, from which a phenotype vector can be extracted:

```
> phen <- data$phen
```

The same can be done for the genotype matrix:

```
> gen <- cbind(data$gen1, data$gen2, data$gen3)
```

4.3 Linear regression

The linear regression is the classical operation performed on genotype-phenotype data. The process consists in estimating a vector of genetic effects $\mathbf{E}_m$ through the least square regression $\mathbf{Y} = (\mathbf{Z}\mathbf{S}_m)\mathbf{E}_m + \mathbf{e}$ (see the section about the NOIA model), $\mathbf{Y}$ being the vector of phenotypes, $\mathbf{Z}$ a matrix containing the genetic information (note that $\mathbf{Z}$ is not gen nor genZ, but it is calculated from them), and $\mathbf{S}_m$) a designed matrix corresponding to model m. In the noia package, the default model considers the mean of the population as a reference point (see the section about implemented models below).

This regression can be performed by the linearRegression() function, which takes at least two parameters: a vector of phenotypes, and a matrix of genotypes (either of type gen or genZ, but not both). A simplistic example inspired from the tiny data set already described above would be:

```
> phen <- c(10,12,9,7,11,16,10,11)
> genot1 <- as.matrix(c(1,2,1,3,2,2,3,NA))
> lr <- linearRegression(phen=phen, gen=genot1)

> lr

Phenotype:
        n= 8  min:  7  max:  16  mean:  10.75
Genotype:
        n= 8 , 1 loci
                Locus 1     1: 0.286     2: 0.429     3: 0.286

   Effects Variances Std.dev  Pr(>|t|)
. 10.75000   0.00000  0.6905 1.986e-05 ***
a -0.50000   0.14286  0.9765   0.63044
d  4.00000   3.91837  1.4916   0.04374 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Variances
        Total (phen)    6.7857
        Residual        2.7245
        Explained       4.0612        (59.8%)
        Genetic         4.0612
```

The phenotype line provides a quick summary of the phenotypic data. Use it to make sure the program understood the data correctly.

The genotype line summarizes the genotype matrix. It provides for each locus the frequency of every genotype.

The following table below is probably the most important piece of information here. With one locus, there are three genetic effects reported. The first one (`.`) is the reference point (here: the mean of the population in the default model). The second line (`a`) is the additive effect, and the third line (`d`) is the dominance effect. The *Variances* column corresponds to the part of genetic variance explained by each effect (note that the variance of the reference point is meaningless). The *Std.dev* column is the standard error of the estimates, which can easily be used to derive an approximate confidence interval, as estimate $\pm$ 1.96 SE. For instance, a good approximation of the confidence interval of the reference point is $[9.4; 12.1]$. The last two columns are provided as part of the standard `R` regression results (probability of being different from 0), it mainly overlaps with the confidence intervals calculated before. It is important to note that the 'statistical significance' of genetic effects here are provided as an indication of the trust that can be put in their values, but they should not be used as a basis to include or exclude loci that were considered as significant by the QTL mapping process. QTL mapping methods are indeed much more powerful and sophisticated than a single linear regression, and can account for *e.g.* multiple testing, which is not done here.

Finally, the regression provides a short summary of the decomposition of genetic variances. The explained variance is the part of the variance attributed to genetic factors (here, the only locus included in the analysis), calculated as the difference between the total variance and the residual variance. The genetic variance is the sum of the variance explained by each genetic effect. Note that, depending on the version of the package `noia`, the variances can be express either at population variances or as sample variances, differing by a factor $n/(n-1)$, which should not make a large difference with large population sizes.

The full decomposition of genetic variance can be obtained through the function `varianceDecomposition()`:

```
> varianceDecomposition(lr)

Variance decomposition:
        Total genetic variance: 4.0612
        Order 1         Total:         4.06122          ( 100.0 %)
                         A          0.14286         ( 3.52 %)
                         D          3.91837         ( 96.5 %)
```

`A` corresponds to the additive variance, and `D` to the dominance variance, in phenotypic units (the unit of variances being the square of the unit of phenotypic measure) and in percentage of the genetic variance. If needed, the percentage of phenotypic variance can be easily calculated from the phenotypic variance provided by the `linearRegression()` function.

There are two important features that can be noticed here: (i) the intercept of the regression (reference point) is exactly equal to the mean phenotype (10.75) , and (ii) the sum of variance components is exactly equal to the part of variance explained by the regression (4.06). These two observations are a consequence of the fact that the model is perfectly orthogonal.

Considering more than one locus multiplies the number of effects, but the interpretation of the analysis remains very similar. For instance, [samepage=true]

```
> phen2loc <- c(10,12,9,7,11,16,10,11,14,12,15,10)
> genot2loc <- cbind(c(1,3,2,1,2,2,1,2,3,2,3,3),
c(1,2,1,3,2,3,2,3,2,1,1,3))
> lr2 <- linearRegression(phen=phen2loc, gen=genot2loc)
Warning message:
In linearRegression(phen = phen2loc, gen = genot2loc) :
  The decomposition of genetic effects is not orthogonal.
```

```
> lr2
Phenotype:
        n= 12  min:  7  max:  16  mean:  11.417
Genotype:
        n= 12 , 2 loci
         Locus 1         1: 0.250        2: 0.417        3: 0.333
         Locus 2         1: 0.333        2: 0.333        3: 0.333

      Effects  Variances Std.dev  Pr(>|t|)
.. 11.3333333  0.0000000  0.7561 0.0006446 ***
a.  1.7831325  1.8326640  0.9827 0.1672193
d.  0.8333333  0.1673360  1.5411 0.6262555
.a -0.5833333  0.2268519  0.9081 0.5663535
aa -0.7108434  0.1707721  1.2629 0.6128392
da  3.5000000  2.1065709  1.7795 0.1438906
.d  0.1250000  0.0034722  1.6346 0.9438580
ad -0.3795181  0.0206715  1.9764 0.8599835
dd -2.0000000  0.1990936  3.4460 0.6023786
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Variances
        Total (phen)        6.6288
        Residual        1.7273
        Explained        4.9015        (74%)
        Genetic         4.7274
```

The code for genetic effects is straightforward: `..` is the reference point, `a.` is the additive effect at the first locus, `.d` is the dominance effect at the second locus, `da` is the dominance-by-additive effect, etc. When more than two loci are involved, higher-order effects are built on the same principle, *e.g.* `a..` is the additive effect of locus 2 out of 4, and `..da` is the dominance-by-additive effect between loci 3 and 4.

```
> varianceDecomposition(lr2)

Variance decomposition:
        Total genetic variance: 4.7274
        Order 1         Total:   2.23032        ( 47.2 %)
                         A       2.05952        ( 43.6 %)
                         D       0.17081        ( 3.61 %)
        Order 2         Total:   2.49711        ( 52.8 %)
                         AA      0.17077        ( 3.61 %)
                         AD      2.12724        ( 45.0 %)
                         DD      0.19909        ( 4.21 %)
```

Each additional level of interactions generate new variance components (here, 3 second-order interaction components A×A, A×D and D×D).

Note that, because of linkage disequilibrium between the two loci, the model is not orthogonal any longer. This can be noticed by (i) the corresponding warning during the regression, (ii) the fact that the reference point slightly differs from the mean phenotype, and (iii) the sum of the variance component is not exactly equal to the total variance minus the residual variance (explained variance).

The function `linearRegression()` provides a number of options, which are detailed in the manual. Among the most useful are those aiming at reducing the complexity of the model, max.level and max.dom. max.level sets the maximum order of interactions that should be included in the model. If `max.level=1`, only marginal effects (*i.e.* first order effects) will be calculated:

```
> lr2b <- linearRegression(phen=phen2loc, gen=genot2loc, max.level=1)
Warning message:
In linearRegression(phen = phen2loc, gen = genot2loc, max.level = 1) :
  The decomposition of genetic effects is not orthogonal.

> lr2b

Phenotype:
        n= 12  min:  7  max:  16  mean:  11.417
Genotype:
        n= 12 , 2 loci
        Locus 1               1: 0.250        2: 0.417        3: 0.333
        Locus 2               1: 0.333        2: 0.333        3: 0.333

     Effects Variances Std.dev  Pr(>|t|)
.. 11.416667  0.000000  0.7468 1.235e-06 ***
a.  1.791924  1.850781  0.9964    0.1152
d.  0.986486  0.234495  1.5627    0.5479
.a -0.250000  0.041667  0.9147    0.7925
.d  0.283784  0.017896  1.6472    0.8681
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Variances
        Total (phen)    6.6288
        Residual        4.2592
        Explained       2.3696        (35.7%)
        Genetic         2.1448
```

Here all epistatic terms have been removed. The max.dom option works in a similar way, setting the maximum level of dominance interactions ; `max.dom=0` is allowed (no dominance at all). `max.level` and `max.dom` can be combined.

Another important option is the `reference=` option, which defines the model that is used. The models that are available in the software are the topic of the next section.

4.4 Genetic models

The package `noia` provides a predefined set of genetic models that can be used to perform the regressions and to compute the variances. This section does not attempt at providing a full description of them, but rather at introducing their main properties.

The default model is `"noia"`. Although not perfectly orthogonal in multi-locus regressions, it provides the best approximation to an orthogonal decomposition of genetic effects so far, and probably constitutes the most logical choice for a statistical approach. The `"G2A"` model also targets orthogonality, but only works in random-mating populations. At the opposite, three purely functional models are provided: `"P1"`, `"P2"` and `"F1"`. The unweighted model, `"UWR"`, has a particular status, since it was primarily defined to compute genetic effects in a fixed mixture of genotypes (all at even frequencies). The two last models, `"Finf"` and `"F2"`, correspond to perfect F_∞ and F_2 populations, and might be of interest when analyzing experimental populations through a breeding procedure expected to produce such patterns.

Model	Code in `noia`	Reference point	Orthogonality
Parental 1	`"P1"`	aa genotype at all loci	never
Parental 2	`"P2"`	AA genotype at all loci	never
Hybrid F_1	`"F1"`	aA genotype at all loci	never
Unweighted model	`"UWR"`	$(G_{aa} + G_{aA} + G_{AA}/3)$	Not in any realistic pop.
F_∞	`"Finf"`	Mid point between aa and AA	Only in perfect F_∞
Intercross	`"F2"`	$(G_{aa} + 2G_{aA} + G_{AA})/4$	Only in perfect F_2
General-2-alleles	`"G2A"`	$G_{aa} \cdot p^2 + 2G_{Aa} \cdot pq + G_{AA} \cdot q^2$	Only if random-mating
NOIA	`"noia"`	$G_{aa}p_{aa} + G_{aA}p_{aA} + G_{AA}p_{AA}$	Any population at linkage equilibrium

Both genetic effects and variance decomposition are affected by the model. Note that reporting values for genetic effects (*e.g.* a or d) as QTL mapping programs usually do is thus meaningless, except when the model that was used is mentioned. Most of the time, this model is probably either Intercross (F_2) or Recombinant Inbred (F_∞), *i.e.* they assume some structure a priori in the population; their accuracy thus depends on the match between the expected structure and the actual population, including variation due to sampling.

The computation of genetic effects obtained from one model into another one does not necessarily involve a new regression. It can be achieved by the change of reference operation, implemented in the function `geneticEffects()`:

```
> geneticEffects(lr2, "P1")
   Effects  Std.dev
..    10.0 2.607896
a.     2.5 1.737626
d.    -2.0 2.619372
.a    -1.5 1.852520
aa    -0.5 1.267406
da     3.5 1.779513
.d     1.5 3.164502
ad    -0.5 1.987244
dd    -2.0 3.446012

> geneticEffects(lr2, "UWR")
      Effects   Std.dev
.. 11.1111111 0.7726609
a.  1.8333333 0.9870321
d.  0.8333333 1.5411035
.a -0.8333333 0.9276034
aa -0.5000000 1.2674056
da  3.5000000 1.7795130
.d  0.3333333 1.6708382
ad -0.5000000 1.9872443
dd -2.0000000 3.4460122
```

Although the interpretation of genetic effects is not particularly easy, especially between models, some general directions can be provided. If, for a given locus, a=+1g in the `"P1"` model, this means that substituting an allele A instead of a at locus 1 has an average effect of +1g when all other loci are of genotype aa. If the additive effect at the same locus is +2g in the `"P2"` model, this means that the very same allelic substitution increases the phenotype by 2g when all other loci are AA. This is some clear evidence for epistasis, since the effect of a substitution changes depending on the genetic background.

If the dominance variance measured in the default `"noia"` model is $12g^2$, and the population was generated by an intercross, one would expect a somehow similar value in the F_2 model. the value obtained by an `"F2"` regression being *e.g.* $11g^2$. This means that the dominance variance, given the genotype-phenotype

relationship estimated from the sample, would be $11g^2$ if the sample was a perfect F2 population. The difference is thus due to the fact that the actual experimental population is not a perfect F_2, either because of sampling effect (a finite population will likely miss the perfect $1/4$, $1/2$, $1/4$ expected proportions) or because of alternative mechanisms (meiotic drive, non-random mating...).

4.5 Genotype-phenotype map

A part of the complexity associated with the analysis of genetic effects (as provided by a QTL mapping routine, or from a `noia` regression) is due to the impact of the underlying model on the effect estimates. Nevertheless, even if different models provide different sets of genetic effects, all of them rely on the same genotype-phenotype relationship. The genotype-phenotype map is thus the common currency to translate estimates into each other, and contains by itself useful hints to understand the consequences of *e.g.* dominance or epistasis on the relationships between genotypes. The genotype-phenotype map can be obtained by the `GPmap()` function:

```
> GPmap(lr)
      G.val  std.err
1  9.535714 1.356109
2 13.035714 1.096958
3  8.535714 1.356109
```

The first column indicates all possible genotypes (1, 2 and 3 only for one locus, with the same code as for the `gen` matrix, *i.e.* 1 for aa, 2 for aA, and 3 for AA), `G.val` gives the estimate of the genotypic value, and `std.err` the standard error of this estimate. As for the genetic effects, a good approximation of the confidence interval can be obtained as estimate $\pm$ 1.96 `std.err`.

With more than one locus, the result is similar, but more combinations are generated:

```
> GPmap(lr2)
   G.val  std.err
11  10.0 2.607896
21  10.5 2.002187
31  15.0 2.259604
12  10.0 2.589849
22  11.0 2.082658
32  13.0 2.242580
13   7.0 2.607896
23  13.5 2.002187
33  10.0 2.259604
```

Genotypes have to be understood as *e.g.* "31: genotype 3 at locus 1, genotype 1 at locus 2". With more than a few loci, the size of the G-P map becomes rapidly very large.

A particularly interesting application of the genotype-phenotype map production is to obtain altered G-P maps, *e.g.* without dominance or without some epistatic components, by computing the G-P map from a regression making use of the options `max.level` or `max.dom`.

4.6 Multilinear regression

The multilinear model, as described in the Model section, is an alternative way to describe and quantify epistasis. The regression is based on the same principles as for the regular genetic effects regression, but assumes that genetic interactions scale with the additive effects of the loci. The `multilinearRegression()` function is very similar to the `linearRegression()` routine, even if the underlying mechanisms are quite different. In particular, the non-linear regression routine can fail, because of a particularly tricky data set, too many loci, or bad automatically calculated starting values. In particular, applying the multilinear regression to large data sets (more than 3 or 4 loci) can be quite tricky.

```
> lr2m <- multilinearRegression(phen=phen2loc, gen=genot2loc,
control=nls.control(maxiter=2000))
Warning message:
In linearRegression(phen = phen, genZ = genZ, reference = reference,  :
  The decomposition of genetic effects is not orthogonal.

> lr2m

Phenotype:
        n= 12  min:  7  max:  16  mean:  11.417
Genotype:
        n= 12 , 2 loci
        Locus 1               1: 0.250        2: 0.417        3: 0.333
        Locus 2               1: 0.333        2: 0.333        3: 0.333

     Effects Variances Std.dev  Pr(>|t|)
.. 11.288505        NA  0.7824 6.945e-06 ***
a.  0.022456        NA  0.3384    0.9493
d.  0.983043        NA  1.5641    0.5528
.a -0.539024        NA  0.8973    0.5700
.d  0.403655        NA  0.7635    0.6160
ee -6.340402        NA 15.2500    0.6920
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Variances
        Total (phen)        6.6288
        Residual        3.4956
        Explained        3.1332        (47.3%)
        Genetic         0
```

Note that an additional option was used (`control=nls.control(maxiter=2000)`) in order to increase the number of steps in the non-linear least squares (`nls`) routine, the default (50 steps) being often too small. The output is very similar to the result of a `linearRegression` function, except that no variance components are provided. The directionality parameter (ε) is represented by the code `ee`. When more than two loci are involved, `ee.` represents the ε between loci 1 and 2, `e.e` between loci 1 and 3, etc.

Here, the estimated value for the directionality parameter is negative (-6.34), which suggests a negative curvature of the genotype-phenotype map: when combining substitutions of positive impact on the trait, effects trend to cancel each other and the resulting phenotype will be less than expected under a purely additive model. Note that the standard error that is reported by the multilinear regression routine is more difficult to interpret than in the linear case, since it stands not only for the expected departure from the 'true' model because of sampling and measurement error, but also from the departure between the 'true' genotype-phenotype map and the multilinear model if the real map is not perfctly multilinear, which is likely to happen.

The `varianceDecomposition()` operation cannot be applied to the result of a multilinear regression. It is however possible to get the estimated genotype-phenotype map:

```
> GPmap(lr2m)
   G.val  std.err
11 12.472234 14.006417
21 10.806468 15.862543
31 12.216670 10.688046
12 11.909367  7.319005
22 11.157437  8.285559
32 11.794005  5.601816
13  7.989668 14.006417
23 13.601516 15.862543
33  8.850643 10.688046
```

Note that the standard errors of estimates are given for information, but they are not accurate: they are calculated under somehow unrealistic assumptions (independence of genetic effects). The multilinear model has less parameters than the full linear genetic models, and the estimated genotype-phenotype map will thus be different.

There are ways to perform a "change of reference" operation in the multilinear framework, but they are not implemented in the package (the function `geneticEffects()` does not work). However, a new regression from a different reference point will provide the expected result.

5 Getting involved

5.1 Questions and bug reports

General questions about `R` and the base packages can be asked on various mailing lists, such as R-help (`https://stat.ethz.ch/mailman/listinfo/r-help`.

Specific questions about the NOIA model and its applications to data should be sent to the corresponding authors of the papers cited in the introduction.

Questions, bug reports, and suggestions about the `noia` package, the implementation of the functions, and the official documentation should be sent to the official maintainers of the package (type `?noia` in `R` after having loaded the `noia` package).

Remarks or comments about this tutorial should be sent to the author, Arnaud Le Rouzic <`lerouzic@legs.cnrs-gif.fr`>.

5.2 Contribution

The `noia` package is released under a free (GLP-2) license and it is possible to install, study, execute, modify and publish the modifications as long as the GPL-2 license is respected (in brief: cite the authors and keep the license unchanged in the modified versions, read the full conditions for more details `http://www.gnu.org/licenses/gpl-2.0.html`).

Contributions, in terms of suggestions, bug reports, bug fixes, and new features are welcome. Note that the patches will of course be reviewed and may not be included in the main project if problematic or too far from the original objectives of the package.

Acknowledgements

Many thanks to José Álvarez-Castro for his remarks and suggestions on this tutorial. I am grateful to Arne Gjuvsland for his contribution to the code, and many users for their remarks and bug reports, including Mihaela Pavlicev, Fran cois Besnier, and Jon Olav Vik. In addition to all these people, Örjan Carlborg and Thomas Hansen are acknowledged for support and helpful discussion.