Population Genetics (52642)

Benny Yakir

1 Introduction

In this course we will examine several topics that are related to population genetics. In each topic we will discuss briefly the biological background and present simple probabilistic models that are used in order to model the biological phenomena. We will attempt to investigate the statistical implication of these models via simulations and mathematical analysis. The R programming system, which will be introduced in class, will be used for the simulations. The needed mathematical background for the mathematical analysis will be provided in class.

The first half of the course will deal with processes in experimental population. The second half will consider models appropriate for natural populations. The topic we intend to explore are (subject to changes):

Elementary facts about the birds and the bees: In this section we will discuss the basics of mating and reproduction.

Basic R: Here we introduce the basic features and structure of the R system.

Inbreeding at a single locus: Here we deal with selfing and brother-sister matings in experimental populations and its effect on heterozygosity of a single locus.

Recombination in experimental genetics: Crossing is initiated between two inbred populations and a pair of loci is considered. Crossovers during meiosis may produce recombinant offspring. The dynamics of the creation of such recombinants will be examined for some crossing designs.

Congenics: Congenic strains are created via a repeated application of selective crossing. In order to investigate the genomic structure of

such strains at multiple loci we will include models for the process of crossovers across the genome.

To add...

2 Elementary Facts About the Birds and the Bees

2.1 The Cell and its Structure

The body of most living things that come to mind when we think of life is made of cells. There are many types of cells, about 200 in human. Nonetheless, these cells share some common properties. In particular, they all have a nucleus. This nucleus contains the instructions for producing the tens of thousands of different types of proteins that make and operate the cell. These instructions are read to special templates, called *m-RNA* and edited within the nucleus. The cells uses these templets to produce the needed proteins according to demand and in a controlled manner. The instructions that are stored in special molecules – DNA – in the nucleus are the genetic material that passes on from generation to generation.

2.2 Chromosomes and DNA

The DNA is a special type of molecule. The molecule is made of repetitive unites of base pairs, which are attached to each other to form a very long chain. There are four types of bases, marked by the letters A, T, C and G. The base A always pairs with the base T and the base C with G. The genetic information for forming and operating the organism is coded using these four bases. The DNA molecules are packed in the molecule in units called *chromosomes*. In human there are 22 pairs of homologous chromosomes (autosomes) and a special pair of sex chromosomes. In female there is a pair of homologous X chromosomes and in male a pair composed of an X chromosome and a Y chromosome. One of the copies in each of the pairs is inherited from the mother and the other copy is inherited from the father.

2.3 Meiosis, the Formation of Gametes, and Crossovers

The process that leads to the formation of the eggs in the female and the sperms in the males is called *meiosis*. At the initiation of this process is a cell with DNA content as described in the previous subsection. At the termination of the process four cells are formed, each with only one copy

of each autosome and with one sex chromosome. During the process the number of chromosomes is doubled and the original cell splits into four.

During the process of splitting the four homolog copies of a chromosome between the four cells *crossovers* may occur. These cross over involve an exchange of intervals of DNA between the two homologous copies. Consequently, the final single copy of an autosome that ends in each of the four cells may be composed of a mosaic in which an interval of DNA that originated from the father may be followed by an interval that originated from the mother and vice versa.

2.4 Reproduction

In the process of reproduction the genetic material of sperm cell is merged with that of an egg cell to form the first cell of child to be. As a conclusion of the discussion that was carried out we may identify that the genetic material of the child is composed of the contribution of the father and the contribution of the mother. These contributions come in complete chromosomal units. Each of these unites, however, are a blend of genetic material contributed from the parent's parents.

3 Introduction to R

R is a freely distributed software for data analysis. In order to introduce R let us quote the first paragraphs from the manual *Introduction to R* by W. N. Venables, D. M. Smith and the R Development Core Team. (The full document, as well as access to the installation of the software itself, are available online at *http://cran.r-project.org/*):

> "R is an integrated suite of software facilities for data manipulation, calculation and graphical display. Among other things it has
>
> - an effective data handling and storage facility,
> - a suite of operators for calculations on arrays, in particular matrices,
> - a large, coherent, integrated collection of intermediate tools for data analysis,
> - graphical facilities for data analysis and display either directly at the computer or on hardcopy, and

- a well developed, simple and effective programming language (called S) which includes conditionals, loops, user defined recursive functions and input and output facilities. (Indeed most of the system supplied functions are themselves written in the S language.)

The term environment is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R is very much a vehicle for newly developing methods of interactive data analysis. It has developed rapidly, and has been extended by a large collection of packages. However, most programs written in R are essentially ephemeral, written for a single piece of data analysis."

R may be obtained as a source code or installed using a pre-compiled code on the Linux, Mackintosh and the Windows operating systems. Programming in R for this book was carried out under Windows. You may find more detailed information regarding the installation of R on the Windows operating system at *http://www.biostat.jhsph.edu/ kbroman/Rintro/Rwin.html*.

After installing R under the `Windows` operating system an icon will be added to the desktop. Double clicking on that icon will open the window of the `R` system, which contains the `R Console` sub-window. We found it convenient to have a separate working directory for each project. It is convenient to copy the `R` icon into that directory and to set the working directory by coping its path (in double quotes) in the appropriate box (`"start in:"`) in the `Shortcuts` slip of the `Properties` of the icon (which can be selected by right-clicking the icon.)

The `R` language is an interactive expression-oriented programming langauge. The elementary commands may consist of expressions, which are immediately evaluated, printed to the standard output and lost. Alternatively, expressions can be assigned to object, which store the evaluation of the expression. In the later case the result is not printed out to the screen. These objects are accessible for the duration of the session, and are lost at the end of the session, unless they are actively stored. At the end of the session the user is prompted to store the entire workspace image, including all objects that were created during the session. If "`Yes`" is selected then the objects used in the current session will be available in the next. If "`No`" is selected then only objects from the last saved image will remain.

Commands are separated either by a semi-colon (;), or by a newline. Consider the following example, which we type into the `R Console` window:

```
> x <- c(1,2,3,4,5,6)
> x
[1] 1 2 3 4 5 6
```

Note that in the first line we created an object named `x` (a vector of length 6, which stores the value 1 ..., 6). In the second line we evaluated the expression `x`, which printed out the actual values stored in `x`. In the formation of the object `x` we have applied the *c*oncatenation function `c`. This function takes inputs and combine them together to form a vector.

Once created, an object can be manipulated in order to create new objects. Different operations and functions can applied to the object. The resulting objects, in turn, can be stored with a new name or with the previous name. In the latter case, the content of the object is replaced by the new content. Continue the example:

```
> x*2
[1]  2  4  6  8 10 12
> x
[1] 1 2 3 4 5 6
> x <- x*2
> x
[1]  2  4  6  8 10 12
```

Observe that the original content of `x` was not changed due to the multiplication by two. The change took place only when we deliberately assigned new values to the object `x`.

Say we want to compute the average of the vector `x`. The function `mean` can be applied to produce:

```
> mean(x)
[1] 7
```

A more complex issue is to compute the average of a subset of `x`, say the values larger than 6. Selection of a sub-vector can be conducted by use of the vector index, which is accessible by the use of square brackets next to the object. Indexing can be implemented in several ways, including the standard indexing of a sequence using integers. An alternative method of indexing, which is natural in many applications, is via a vector with logical `TRUE/FALSE` components. Consider the following example:

```
> x > 6
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
> x[x > 6]
[1]  8 10 12
> mean(x[x > 6])
[1] 10
```

Observe that the vector `x > 6` is a logical vector of the same length as the vector `x`. Only the components of `x` parallel to the components with a `TRUE` value in the logical indexing vector are selected. In the last line of the example the resulting object is used as the input to the function `mean`, which produces the expected value of 10.

For comparison consider a different example:

```
> x*(x > 6)
[1]  0  0  0  8 10 12
> mean(x*(x >6))
[1] 5
```

In this example we multiplied a vector of integers `x` with a vector of logical values `(x > 6)`. The result was a vector of length 6 with zero components where the logical vector takes the value `FALSE` and the original values of `x` where the logical value takes the value `TRUE`. Two points should be noted. First, observe that R can interpret a product of a vector with integer components and a vector with logical components in a reasonable way. Standard programming languages may have produced error messages in such a circumstance. In this case, R translates the logical vector into a vector with integer values — one for `TRUE` and zero for `FALSE`. The outcome, a product of two vectors with integer components, is a vector of the same type. The second point to make is that multiplication of two vectors is conducted term by term. It is not the inner product between vectors. A different operator is used in R in order to preform inner products.

Next let us try to program in R functions that simulate the processes meiosis and mating. Imagine a given locus on a given chromosome with two possible variate forms (*alleles*). One is denoted by "A" and the other by "a". We have in mind aa animal, say a laboratory mouse, that produces gametes. Assume that the paternal allele of the mouse is "A" and the maternal allele is "a".

```
> n <- 9
> pat <- rep("A",n)
```

```
> mat <- rep("a",n)
> pat
[1] "A" "A" "A" "A" "A" "A" "A" "A" "A"
> mat
[1] "a" "a" "a" "a" "a" "a" "a" "a" "a"
> mode(pat)
[1] "character"
```

Observe that `pat` and `mat` are vectors of character strings. They are of length 9. The function `rep` is useful to produce repetitions of patterns. Combining it with the function `seq`, that forms regular sequences is useful at times.

The gametes that are produced by the process of meiosis may be of either of the types at the given locus. According to Mendel's first law of segregation, the probabilities of the two types are even:

```
> from.mat <- rbinom(n,1,0.5)
> offspring <- pat
> from.mat==1
[1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
> offspring[from.mat==1]
[1] "A" "A" "A" "A" "A" "A" "A"
> mat[from.mat==1]
[1] "a" "a" "a" "a" "a" "a" "a"
> offspring[from.mat==1] <- mat[from.mat==1]
> offspring
[1] "a" "a" "A" "a" "a" "a" "a" "A" "a"
> 2:6
[1] 2 3 4 5 6
> offspring[2:6]
[1] "a" "A" "a" "a" "a"
> offspring[-(2:6)]
[1] "a" "a" "A" "a"
```

In the final part of this demonstration we use integers to index components of the vector. The minus sign is used in order to identify the indices we would like to *exclude*.

It would be convenient to wrap the code that produces random gametes in a function. R functions produced with the aid of the function `function`. The arguments of the produced function are identified in the brackets. The function evaluates the expression that follows. Composite expressions can be combined together by placing them within curly brackets. The output

of the function may be specified with the function `return`. Otherwise, the function returns the evaluation of the expression.

```
> meiosis <- function(GF,GM)
+ {
+     from.GM <- rbinom(length(GF),1,0.5)
+     GS <- GF
+     GS[from.GM==1] <- GM[from.GM==1]
+     return(GS)
+ }
> meiosis(pat,mat)
[1] "A" "a" "a" "a" "A" "a" "A" "a" "A"
```

The the process of mating a father mouse is donating its gametes (sperm) to the mother. These gametes merge with the mother's gametes (eggs) to produce the offsprings.

```
> male <- list(pat=rep("A",n),mat=rep("a",n))
> female <- list(pat=rep("a",n),mat=rep("a",n))
```

We consider a father, which is heterozygote at the given locus and a mother, which is homozygote at that locus. An animal is represented by a list, which contains the two grand-parental alleles. List in R are a special type of vector. Each entry to a vector of type "`list`" can store any type of object, regardless of the type of objects in the other entries. Here each entry stores a vector. The entries in this example are given names, which can then be used in order to refer to the entry. We demonstrate that using the function "`cross`", which simulates the formation of offspring by crossing a mother mouse with father mouse:

```
> cross <- function(fa,mo)
+ {
+     pat <- meiosis(fa$pat,fa$mat)
+     mat <- meiosis(mo$pat,mo$mat)
+     return(list(pat=pat, mat=mat))
+ }
```

Running the function produces:

```
> cross(male,female)
$pat
[1] "A" "a" "a" "A" "A" "a" "a" "A" "A"
```

```
$mat
[1] "a" "a" "a" "a" "a" "a" "a" "a" "a"
```