Introduction to R: practical

Lewis Spurgin

May 2017

Introduction

The aim of this practical is to get you comfortable using R and R studio. By the end of this practical, you ahould be able to explore data frames, write a short program to perform some basic data processing, and plot and test some results. We will also have our first go at reading data into R. Some general tips:

- Write all your commands in a SCRIPT and run them from there.
- Annotate your script well (using `#`). No amount of annotation is too much if it helps you understand what you're doing.
- Use R's help (e.g. `?mean`) if you are not sure what a function does
- Google is your friend

The data

Scientists have taken blood samples from bearded finches captured from mainland and island populations. They have genotyped all the samples at a single micorsatellite locus, at which they found two alleles of 100bp an 102bp, respectively. They wish to know whether, at this locus, the mainland and island populations have different levels of genetic diversity.

Instructions

Our aim is to calculate *observed heterozygosity* (i.e. the proportion of heterozygotes) in each of the populations in our dataset. We will plot these proportions, and test whether levels of heterozygosity differ between the two populations.

If you think you can do this without guidance, please go ahead and do so. Otherwise, step-by-step instructions are provided below.

Step 1. Set the working directory, read in and explore the data.

Set your working directory to the "Lectures/Intro_to_R" folder. Note how the command for changing the working directory in R differs to bash and python.

```r
setwd("path/PopGenBerlin/Lectures/Intro_to_R")
```

Minimise R studio for a moment, and open up your file explorer. Find the *Intro_to_R/Data* folder for this lecture, and open the file *Microsat_data.csv*. You can open this in a spreadsheet editor (e.g. excel), or in your text editor. You should be able to see tht the data has four columns, and lots of rows. Each row is an individual. The columns correspond to individual ID, population ID, and the size (in bp) of the two alleles at the microsatellite locus. This is a standard way of storing microsatellite genotype data.

We are now going to data read this data into R using the function `read.csv()`, and assign it to an object name. We will use the name `dd`. Go back to R studio. In your R script, type the following:

```r
dd <- read.csv("Data/Microsat_data.csv")
```

CLick on the “environment” tab in the top-right pane of R studio. The object `dd` should now be listed. If so, the data has been successfully loaded into R.

We can explore our data by typing some functions into our script and running them. We will use the functions `head()` and `str()`.

```r
head(dd)
```

```
##   Ind_ID      Pop Allele_1 Allele_2
## 1  Ind_1 Mainland      102      102
## 2  Ind_2 Mainland      100      100
## 3  Ind_3 Mainland      102      102
## 4  Ind_4 Mainland      100      100
## 5  Ind_5 Mainland      100      102
## 6  Ind_6 Mainland      100      102
```

```r
str(dd)
```

```
## 'data.frame':    250 obs. of  4 variables:
##  $ Ind_ID  : Factor w/ 250 levels "Ind_1","Ind_10",..: 1 112 174 185 196 207 218 229 240 2 ...
##  $ Pop     : Factor w/ 2 levels "Island","Mainland": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Allele_1: int  102 100 102 100 100 100 100 102 102 100 ...
##  $ Allele_2: int  102 100 102 100 102 102 102 102 102 100 ...
```

The function `head()` can be very useful when you want a quick look data frames with many rows. But `str()` gives us all the same information and more. From the output above, you should be able to answer the following:

- How many individuals are there in your dataset?
- How many populations are there?
- How has R stored the microsatellite allele data?
- How has it stored the Individual and Population ID data?

Before we go further, take a look at the help sheet for the `read.csv()` function, by typing `?read.csv` into the console. You will see that there are lots of options within this function - for now, scroll down the list of arguments and read the entry for `as.is`. Have a think and discuss with a neighbour what you think this means. Compare the following to what you ran previously.

```r
dd <- read.csv("Data/Microsat_data.csv",as.is = T)
str(dd)
```

```
## 'data.frame':    250 obs. of  4 variables:
##  $ Ind_ID  : chr  "Ind_1" "Ind_2" "Ind_3" "Ind_4" ...
##  $ Pop     : chr  "Mainland" "Mainland" "Mainland" "Mainland" ...
##  $ Allele_1: int  102 100 102 100 100 100 100 102 102 100 ...
##  $ Allele_2: int  102 100 102 100 102 102 102 102 102 100 ...
```

What is different here? What is the difference between a character and a factor? We will go into this in more detail in the next session. Note also the `$` symbols before each of the variable names. We can call up individual variable using this notation.

```
dd$Pop
```

```
##   [1] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##   [7] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [13] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [19] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [25] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [31] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [37] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [43] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [49] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [55] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [61] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [67] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [73] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [79] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [85] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [91] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
##  [97] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [103] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [109] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [115] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [121] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [127] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [133] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [139] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [145] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [151] "Mainland" "Mainland" "Mainland" "Mainland" "Mainland" "Mainland"
## [157] "Mainland" "Island"   "Island"   "Island"   "Island"   "Island"
## [163] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [169] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [175] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [181] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [187] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [193] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [199] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [205] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [211] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [217] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [223] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [229] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [235] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [241] "Island"   "Island"   "Island"   "Island"   "Island"   "Island"
## [247] "Island"   "Island"   "Island"   "Island"
```

We can also use square bracket notation to call up individual rows and column. Using this notation, get R to print out:

- the 45th row
- the 2nd column
- the value of *Allele_1* for the 22nd individual

Step two - define our genotypes

In our dataset we have separate information for each of the two alleles found in each individual. For example, we can see from when we ran the `head()` function that the first individual in our dataset has an allele of length 102 for its first allele, and an allele of length 102 for its second allele. This individual is therefore a homozygote at this microsatellite locus. We can tell R to give use this information to tell us about heterozygosity. But first, we need to get an understanding of logical operators in R. Try typing the following commands directly into your terminal (I know we said not to do this, but it's ok here!):

```
4 < 3
4 > 3

4 = 3
4 == 3 #gives us an error. Why?

x = 4
x == 4

"cat" == "dog"

x <- "cat"
x == "cat"
```

Hopefully you have figures out that a single equals sign and double equals sign mean very different things to R. A single equals sign generally works in the same way as the `<-` operator (with a few subtle differences), whereas a double equals sign asks a logical question, and returns a value of `TRUE` or `FALSE`.

So coming back to our microsatellite data, we can use the `==` operator to ask whether an individual is a homozygote, by asking whether the variables `Allele_1` and `Allele_2` are the same.

```
# Get the data for the first individual and have a look at it
ind1 <- dd[1,]
ind1
```

```
##   Ind_ID      Pop Allele_1 Allele_2
## 1  Ind_1 Mainland      102      102
```

```
# Now ask whether Allele 1 and Allele 2 are identical
ind1$Allele_1 == ind1$Allele_2
```

```
## [1] TRUE
```

Because R works so nicely with vectors, we can do this for every individual in our dataset, as follows:

```
dd$Allele_1 == dd$Allele_2
```

```
##   [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE
##  [12] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  [23]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE
##  [34]  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
##  [45] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
##  [56]  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
##  [67]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE
##  [78]  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE
##  [89]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## [100] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [111]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
## [122]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE
## [133]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE
## [144]  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
## [155] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## [166]  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [177]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## [188] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## [199]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [210]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [221]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [232]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE
## [243] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

This gives us 250 `TRUE` or `FALSE` values, and tells us whether each individual in our dataset is a homozygote or not. Now all we need to do is turn this into a column of our data frame, indicating whether each individual is a homozygote or a heterozygote. We can do this using the `ifelse()` function. Check out what this function does using `?ifelse`

```r
#Add a Heterozygosity column
dd$Het <- ifelse(dd$Allele_1 == dd$Allele_2,"Homozygote","Heterozygote")

#Check that it's worked
head(dd)
```

```
##   Ind_ID      Pop Allele_1 Allele_2          Het
## 1  Ind_1 Mainland      102      102   Homozygote
## 2  Ind_2 Mainland      100      100   Homozygote
## 3  Ind_3 Mainland      102      102   Homozygote
## 4  Ind_4 Mainland      100      100   Homozygote
## 5  Ind_5 Mainland      100      102 Heterozygote
## 6  Ind_6 Mainland      100      102 Heterozygote
```

We can see that the homozygotes and heterzygotes are being correctly identified. Now we can do some calculations and make a plot.

Step three - calculate and plot genotype frequencies

Now we want to calculate the proportion of heterozygotes in each of our two populations. To do this, we need to know i) the number of heterozygotes in each population and ii) the total number of individuals in each population. We can get both of these pieces of information using the function `table()`.

```r
table(dd$Het) #Number of each genotype
```

```
## 
## Heterozygote   Homozygote 
##           85          165
```

```
table(dd$Pop) #Number of individuals in each population
```

```
##
##   Island Mainland
##       93      157
```

```
table(dd$Het,dd$Pop) #Both combined
```

```
##
##                Island Mainland
##   Heterozygote     19       66
##   Homozygote       74       91
```

We can assign the output of the `table()` function to an object. Then using this we can extract subsets of tata and perform calculations using them. Using this approach, we can divide the number of heterozygotes by the sample size to calculate observed heterozygosity.

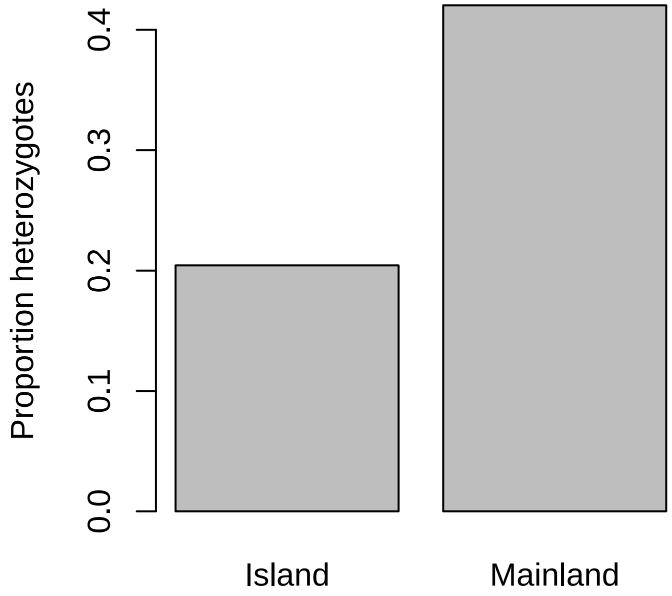
```
#Get the number of heterozygotes and the sample size for each population
het_counts <- table(dd$Het,dd$Pop)["Heterozygote",] #pulls out the heterozygote row
samplesizes <- table(dd$Pop) #Calculate sample sizes

#Now divide the two together
het_freqs <- het_counts/samplesizes
het_freqs
```

```
##
##    Island  Mainland
## 0.2043011 0.4203822
```

Now we can plot the frequencies. We will get into plotting in much more detail later in the course. For now, we will use the function `barplot()`.

```
barplot(het_freqs,ylab = "Proportion heterozygotes")
```



Finally, we can test whether the proportion of heterozygotes differs between mainland and island population using a Chi-squared test (`?chisq.test`).

```r
chisq.test(dd$Het,dd$Pop)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  dd$Het and dd$Pop
## X-squared = 11.208, df = 1, p-value = 0.0008143
```

What are you able to conclude from this? Is the effect significant? Is it in the expected direction? What is lacking from this data?

Finished? Some more challenging tasks

- Calculate and plot the frequency of the two alleles in each population.
- Calculate expected genotype frequencies for each population, according to Hardy-Weinberg. Compare your observed and expected frequencies using chi-squared tests.
- I simulated the microsatellite dataset for this practical in R. However, I did a dreadful job of annotating the script that I used to generate this dataset. Go through the script "Microsat_data.R", figure out what is being done with each line of code, and annotate the script accordingly.
- Simulate your own microsatellite dataset, either based on my approach, or by developing your own code.