Statistical methods in genetic relatedness and pedigree analysis

NORBIS course, Oslo, June 2022

Magnus Dehli Vigeland and Thore Egeland

Exercise set II. Introduction to R and the ped suite

Introduction

The aim of these exercises is to acquaint you with the *ped suite*, a collection of packages for pedigree analysis in R. The exercises below mainly use **pedtools** (manipulating and plotting pedigrees) **verbalisr** (relationship descriptions) and **pedprobr** (probabilities on pedigrees). You rarely need to know which package a function actually belongs to, but one way to find out is by looking at the bottom of its help page (e.g., `?nuclearPed`).

Exercise II-1 (Getting started)

The purpose of this exercise is to ensure that you have installed the *ped suite* correctly, and to establish some good habits when using R/Rstudio. In particular, it is vital to keep your commands in a script rather than executing everything directly in the console. In short, scripting makes your life (and mine!) a lot easier!

a) Open a clean Rstudio session and start a new R script. (Hint: *File* menu → *New File* → *R script*.)

b) At the top, write the following:

```
# Exercise set II. Intro to R/pedsuite.
```

The hashtag `'#'` tells R that this is just a comment, not to be executed. It is good practice to add comments to your code, explaining to the reader (especially future you!) what is going on.

c) On a new line, write the following command and execute it by pressing ctrl+enter (Windows) or cmd+enter (Mac).

```
library(pedsuite)
```

If successful, this loads the core ped suite packages, including those mentioned in the introduction. If, instead, you got an error message like `there is no package called pedsuite`, you have to install it first: *Tools* menu → *Install Packages*.

d) To check that everything works, enter this command on a new line in the script and execute with ctrl+enter. The result should be a simple pedigree plot.

```
plot(nuclearPed())
```

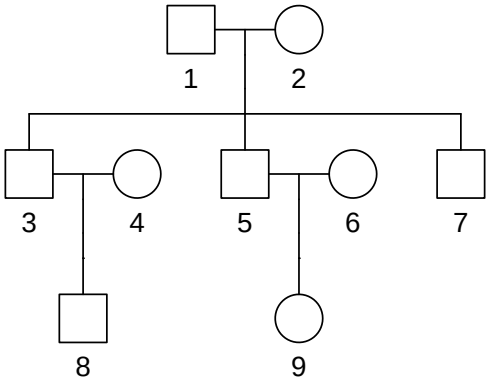
e) Save the script to a file named `exer1.R` and put it in a folder named `NORBIS-relatedness-2022` (or something similar of your own choosing).

f) Make this folder the current working directory.
Hint: *Session* menu → *Working Directory* → *To Source File Location*.

Congratulations, you are now all set to solve the remaining exercises!

Exercise II-2 (Building pedigrees)

This exercise illustrates a typical workflow for creating pedigrees in R. To keep track of the process, I recommend running `plot(x)` after each line of code.



a) Use the code below to create the pedigree in Figure 1. *Plot the pedigree after each line.*

```
x = nuclearPed(nch = 3)
x = addSon(x, 3)
x = addDaughter(x, 4)
x = relabel(x, "asPlot")  # relabel according to plotting order
```

b) The **pedsuite** works well with *pipe* operator |> recently introduced in R. The pipe makes the result of the previous command become the first argument of the next. For example, the following code is equivalent to the one in a). Verify this by running the following code and plotting the result.

```
x = nuclearPed(nch = 3) |>
  addSon(3) |>
  addDaughter(4) |>
  relabel("asPlot")
```

c) Add a child with parents 8 and 9, and plot the result.

Exercise II-3 (Built-in basic pedigrees)

Use R to create and plot the following relationships. Each pedigree can be created with a single command. You may want to consult the help pages, e.g., `?linearPed`.

a) Aunt – nephew. (`avuncularPed`)
b) Great-grandparent – great-grandchild. (Hint: `linearPed()`.)
c) First cousins once removed. (Hint: `cousinPed()`.)
d) Half second cousins. (Hint: `cousinPed()`.)

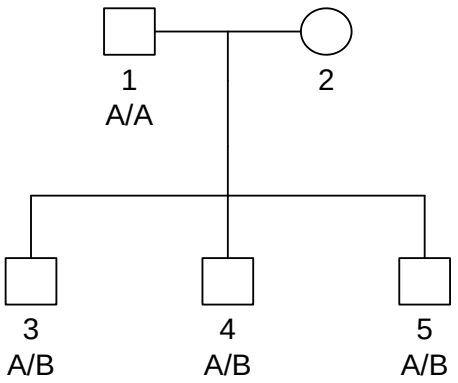
Exercise II-4 (Inbred pedigrees)

a) Use R to create and plot a pedigree showing father-daughter incest.
b) Use R to create and plot a pedigree showing brother-sister incest.

Exercise II-5 (Genotypes and plot options)

a) Use the following code to create the pedigree below.

```
x = nuclearPed(3)
x = addMarker(x, geno = c("A/A", NA, "A/B", "A/B", "A/B"))
plot(x, marker = 1)
```



b) With `x` as above, try the following commands and make sure you understand how the different options work. The help page for pedigree plotting contains additional information: type `?plot.ped` to open it.

```
plot(x, title = "MY PED", aff = 3:5, deceased = 1:2)
plot(x, hatched = 3:5, col = list(red = 1:2, blue = 3:5))
plot(x, marker = 1, showEmpty = T, missing = "*", sep = ":")
plot(x, marker = 1, cex = 2, labs = NULL)
```

Exercise II-6 (Pedigree likelihood)

We will calculate the likelihood of the pedigree in the previous exercise, assuming that the marker is an autosomal SNP in Hardy-Weinberg equilibrium, with allele frequencies $P(A) = 0.8$ and $P(B) = 0.2$.

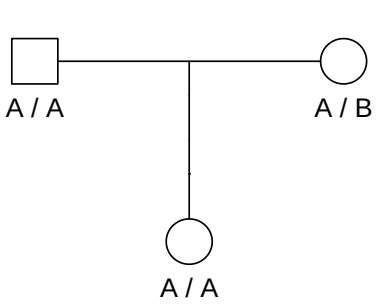
a) With `x` as in the previous exercise, use `afreq(x, marker = 1)` to inspect the default allele frequencies. What are they?

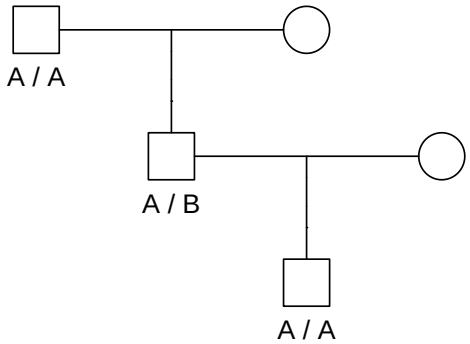
b) Set the correct allele frequencies and compute the likelihood as follows.

```
x = setAfreq(x, marker = 1, afreq = c(A = 0.9, B = 0.1))
likelihood(x, marker = 1)
```

Exercise II-7 (More likelihoods)

In the pedigrees below, the genotypes are for an autosomal SNP in Hardy-Weinberg equilibrium. We assume absence of mutations. Compare your answers with those you got by manual calculation in Exercise set I.



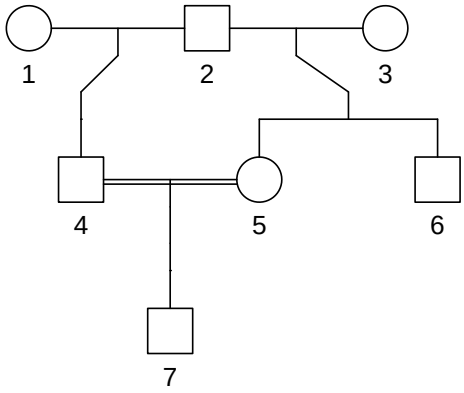


a) Use R to compute the likelihood of the left-most pedigree, if $p_A = p_B = 0.5$.

b) Use R to compute the likelihood of the right-most pedigree, if $p_A = 0.9$ and $p_B = 0.1$.

Exercise II-8 (Creating and loading a ped file)

a) Create the pedigree shown below using QuickPed (https://magnusdv.shinyapps.io/quickped/). Store the pedigree as a ped file named "quick-1.ped" in the folder created in Exercise II-1.



b) Load the pedigree into R using `readPed()` and plot it to check that it looks correct.

```
x = readPed("quick-1.ped")
plot(x)
```

c) Describe the relationship between 6 and 7. Hint: `verbalise(x)`.

Exercise II-9 (X-linked recessive inheritance)

The shaded male in the pedigree below has hemophilia A, an X-linked recessive disease. None of the other family members are affected, but the female cousin (8) is worried that she might be carrying the disease allele. We assume that the disease allele D has frequency 0.001 in the population, and that HWE holds. We also assume that no mutations at the disease locus arose within the pedigree.

a) Write down all the genotypes you can deduce from the given information.

b) What is the probability that individual 8 is a carrier of the disease allele? Were the assumptions about frequency and HWE needed to answer this?

c) The information about the grandfather is limited, and it is possible that he may have had the disease. Explain how this complicates the computation of the carrier probability.

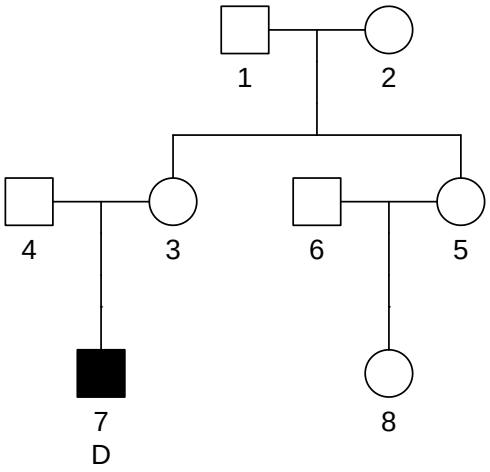
d) The function `oneMarkerDistribution()` can be used to to compute such probabilities. The following code is a good starting point in this case:

```
# Create pedigree
ped = cousinPed(1) |>
  swapSex(c(4, 6, 8)) |>
  addMarker(afreq = c(D = 0.001, N = 0.999), chrom = "X") |>
  setGenotype(marker = 1, id = 7, geno = "D")

# Plot
plot(ped, marker = 1, aff = 7)

# Genotype distribution
oneMarkerDistribution(ped, id = 8, partialmarker = 1)
```

What is the reported probability that 8 is a carrier? Discuss the output and possible weaknesses of this calculation.



Bonus exercises

Exercise II-10 (Merging pedigrees)

A powerful technique for building complex pedigrees is to create separate branches and "glue" them together with the `mergePed()` function. Here we work through an example with a family trio in which both the father and the mother have first-cousin parents.

a) We start by making the three parts separately.

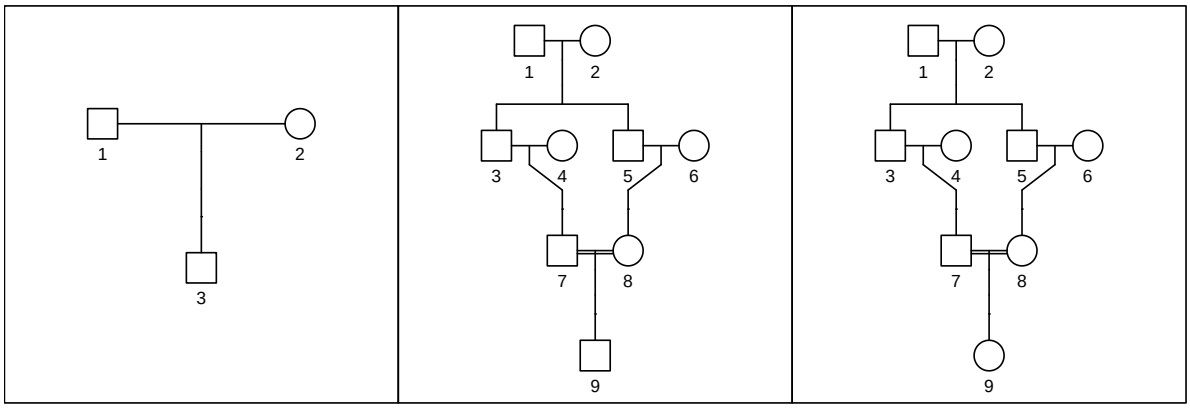
```
# Trio
x = nuclearPed(1)

# Fathers pedigree: Boy with first cousin parents
fa = cousinPed(1, child = T)

# Mothers pedigree: Girl with first cousin parents
mo = swapSex(fa, 9)
```

b) To prepare the merge, plot all three parts. A handy tool for this is `plotPedList()`, which is designed to plot multiple pedigrees together.

```
plotPedList(list(x, fa, mo))
```



c) For the first merge we want to identify individual 1 in the first pedigree with 9 in the second. Use the following command to do this, and plot the result afterwards.

```
x2 = mergePed(x, fa, by = c("1" = "9"))
plot(x2)
```

Why have some of the labels changed?

d) Finish the pedigree by gluing the mother's pedigree onto `x2`.
Tip: Add `relabel = T` to the `mergePed()` command to label everyone according to plot order.

e) (For pipe enthusiasts) Merge `x`, `fa` and `mo` in a single command using the pipe operator.

Exercise II-11 (Quadruple second cousins)

Create a pedigree with quadruple second cousins.

Hint: Make the fathers double first cousins, and the same for the mothers. Use `doubleCousins()` twice and glue the parts with `mergePed()`. The final plot may look confusing, but you should verify the relationship with `verbalise()`.