Young Investigator Day 2024

Genetic relatedness

Teacher

Magnus Dehli Vigeland, PhD

Dept of Forensic Sciences, OUH

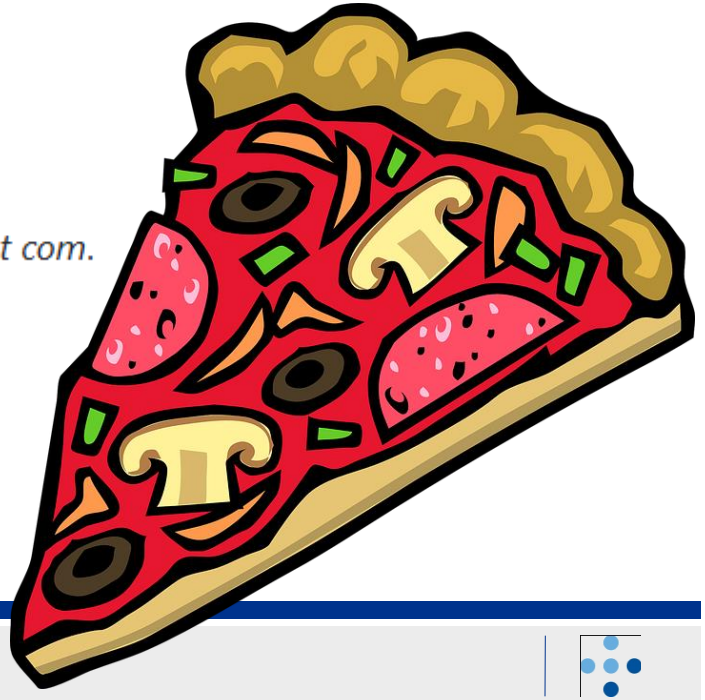
Home page
https://magnusdv.github.io/pedsuite/articles/web_only/course-yid2024.html

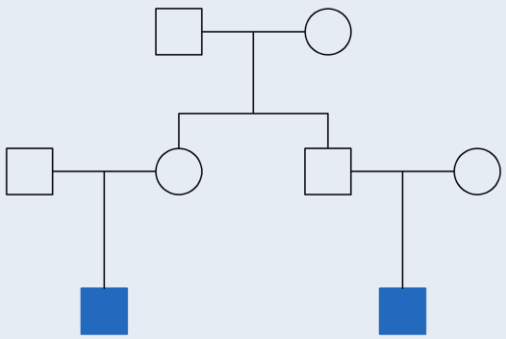
Schedule

The workshop is run as a half-day course, from 12 to 16, with the following (tentative) schedule:

- 12:00–13:00 Lecture 1. **Introduction to pedigrees, QuickPed and R**
- 13:00–13:45 Exercises
- 13:45–14:00 *Break*
- 14:00–15:00 Lecture 2. **Measures of relatedness**
- 15:00–15:45 Exercises
- 15:45–16:00 Summary and discussion

Comments and questions may be sent to *magnusdv at gmail dot com*.





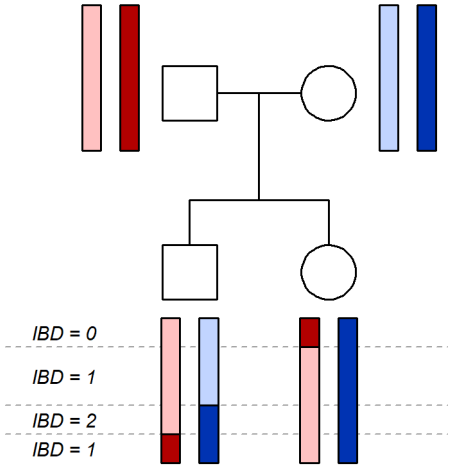
Lecture 1:
Pedigrees and relationships

Magnus Dehli Vigeland

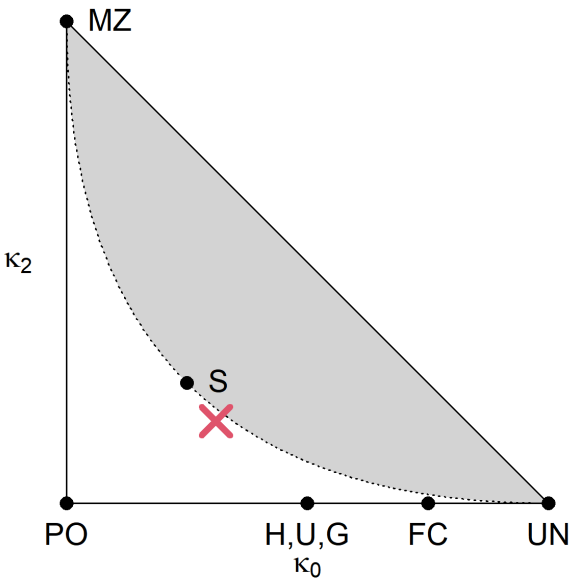
Young Investigator Day 2024

Outline

- Now: *Pedigrees*
  - Conventions and terminology
  - QuickPed
  - Crash course in R/pedsuite

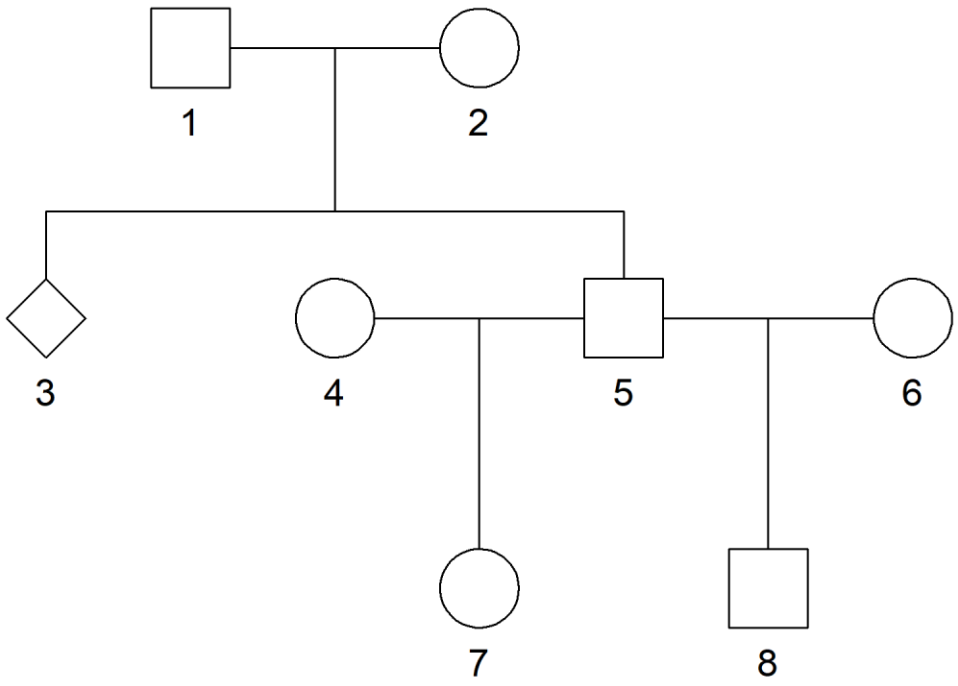


- Later today: *Measures of relatedness*
  - Identity by descent (IBD)
  - Coefficients of kinship and inbreeding
  - The relatedness triangle
  - Realised relatedness



Pedigrees

Conventions and terminology

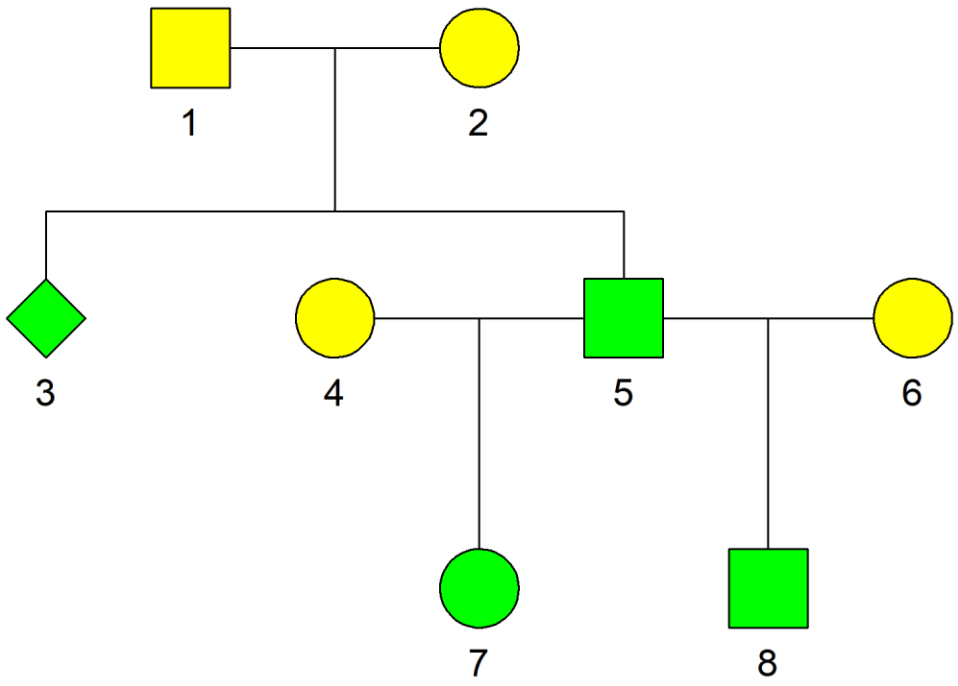


= male

= female

= unknown

Conventions and terminology



= male

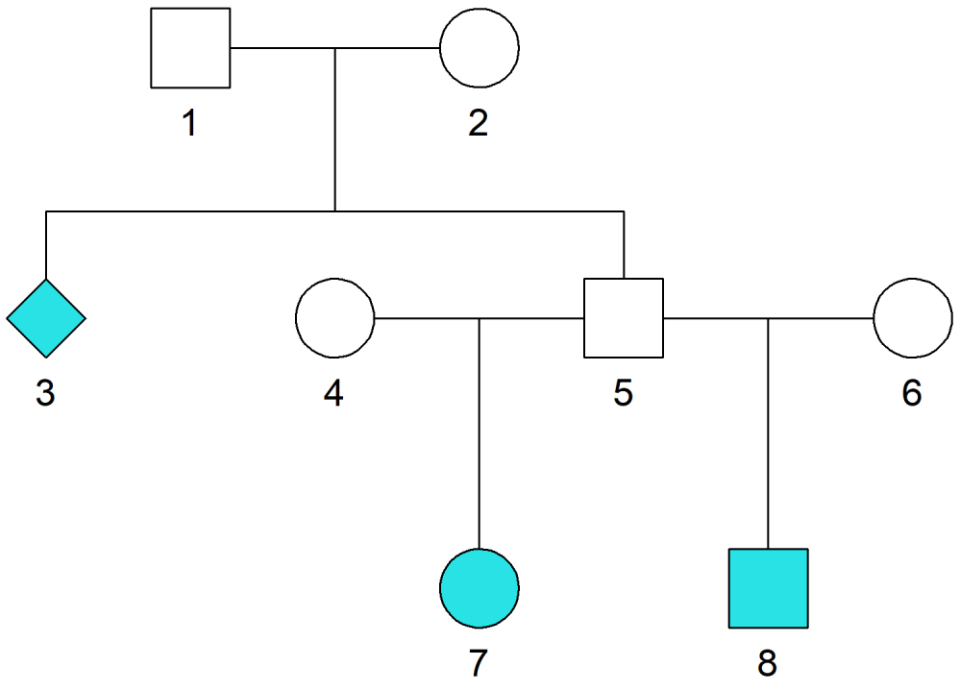
= female

= unknown

Founders
No parents included in the pedigree

Nonfounders
Parents are included

Conventions and terminology



= male

= female

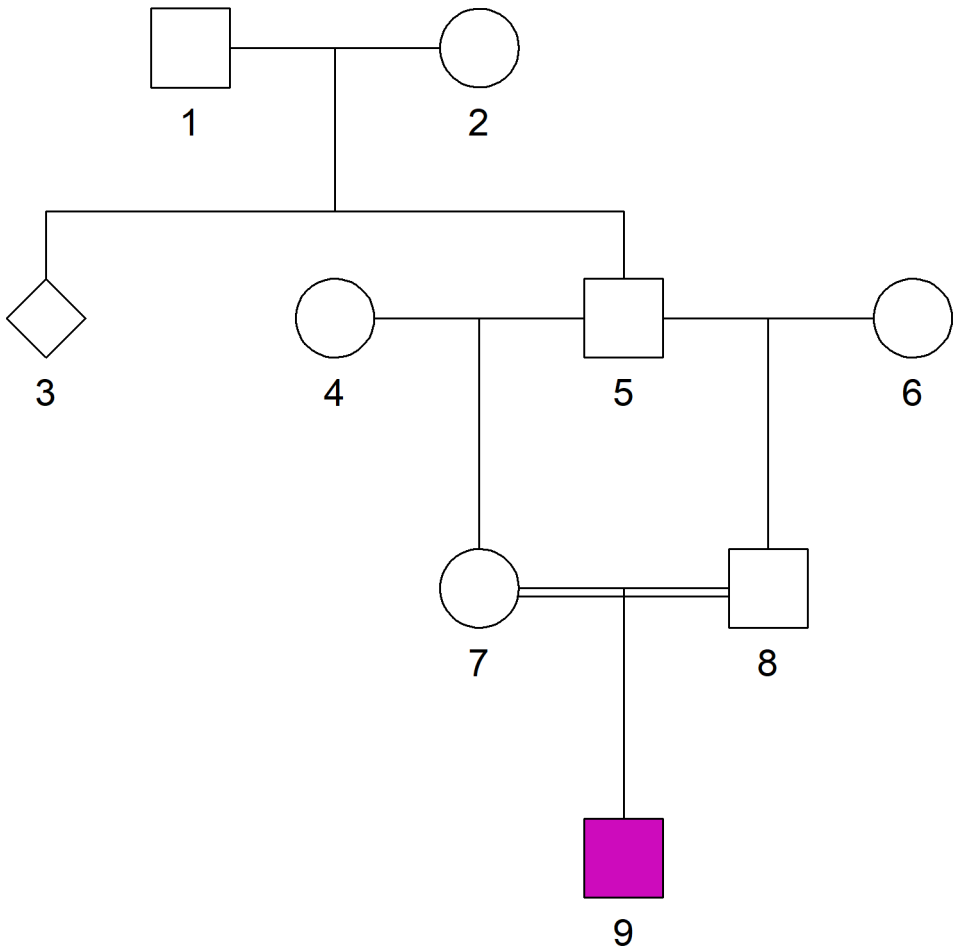
= unknown

Founders
No parents included in the pedigree

Nonfounders
Parents are included

Leaves
No children included

Conventions and terminology



= male

= female

= unknown

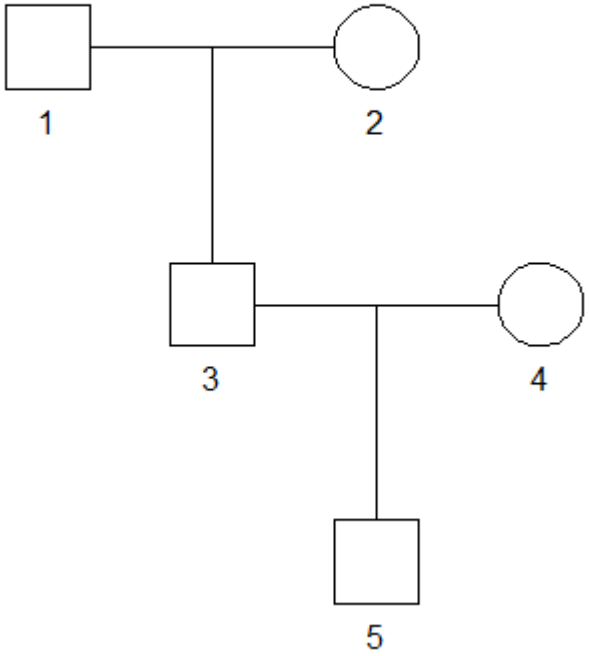
Founders
No parents included in the pedigree

Nonfounders
Parents are included

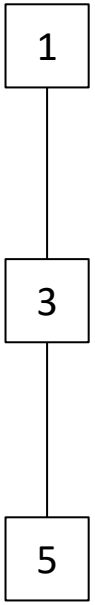
Leaves
No children included

Inbred
Children of related parents

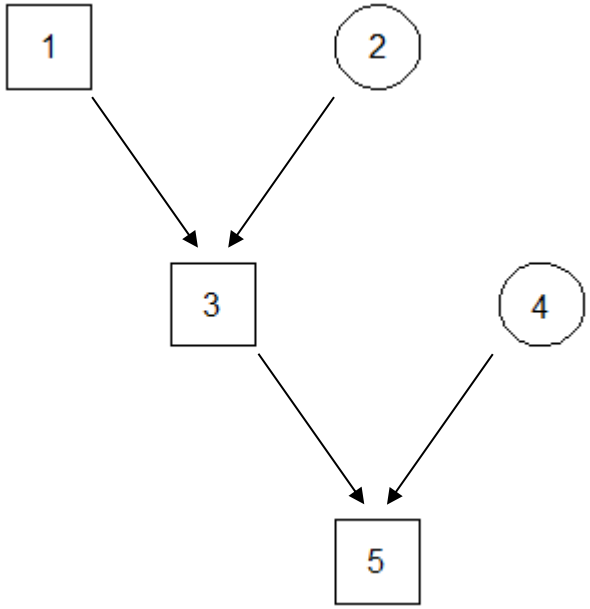
Alternative ways of drawing pedigrees



Standard



Simplified

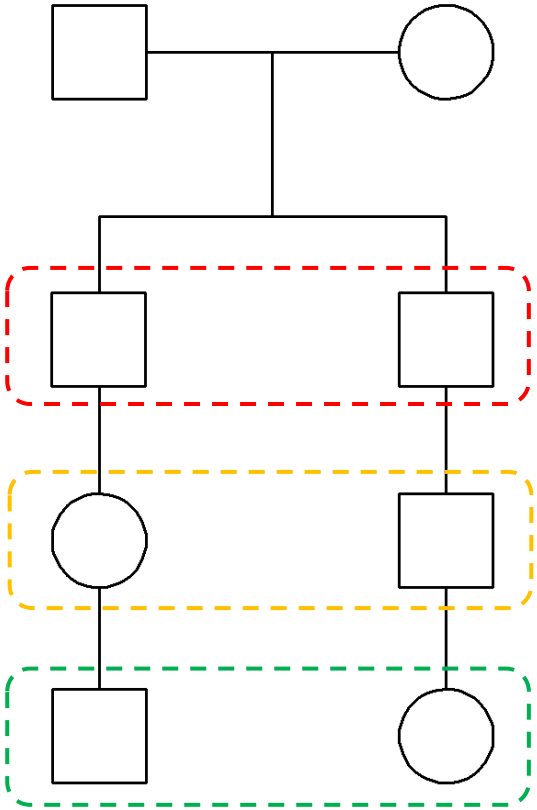


Directed acyclic graph

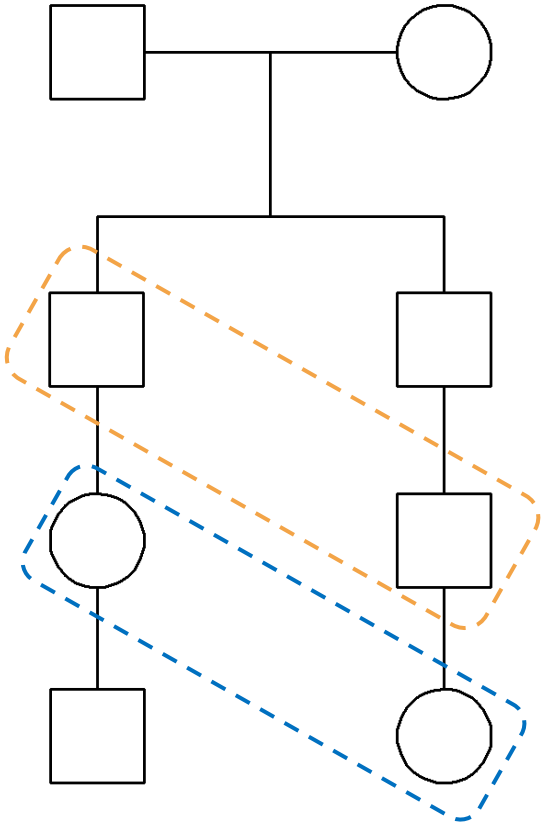
Some common relationships

(and some less common...)

Full relationships



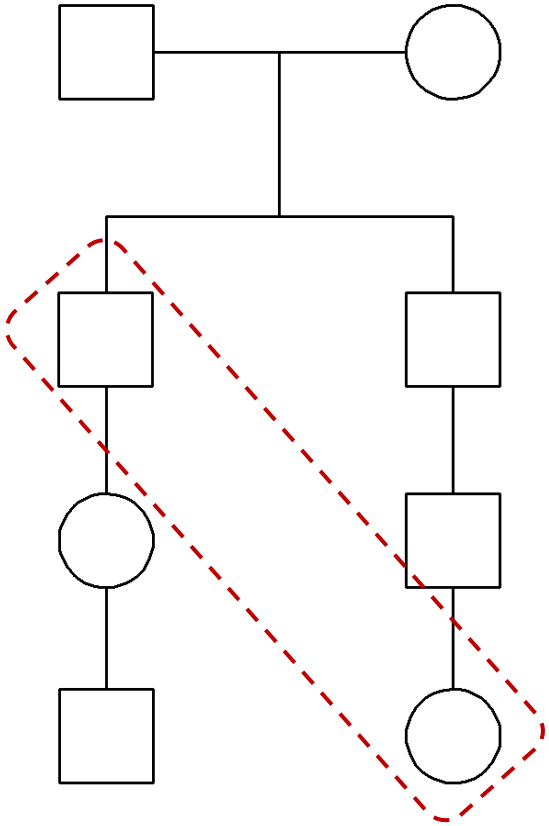
Full relationships



Uncle - nephew
(*avuncular*)

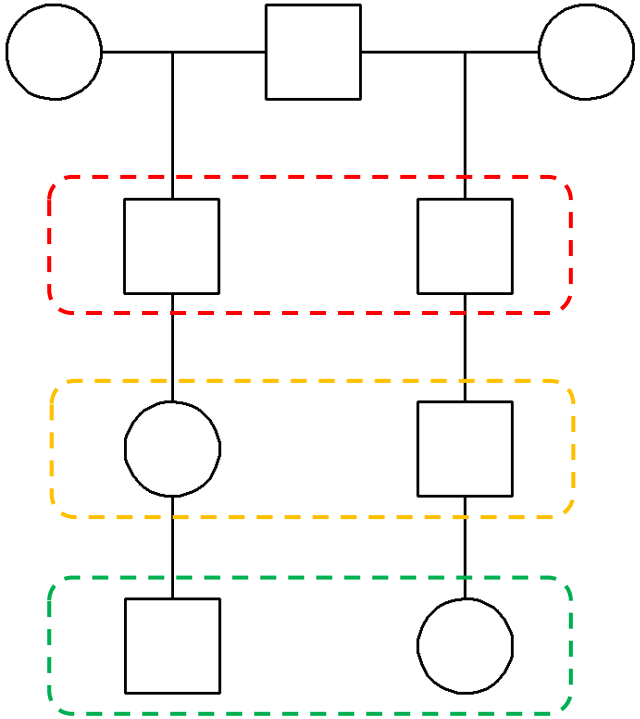
First cousins
once removed

Full relationships



Grand-uncle
(or *great uncle*)

Half relationships

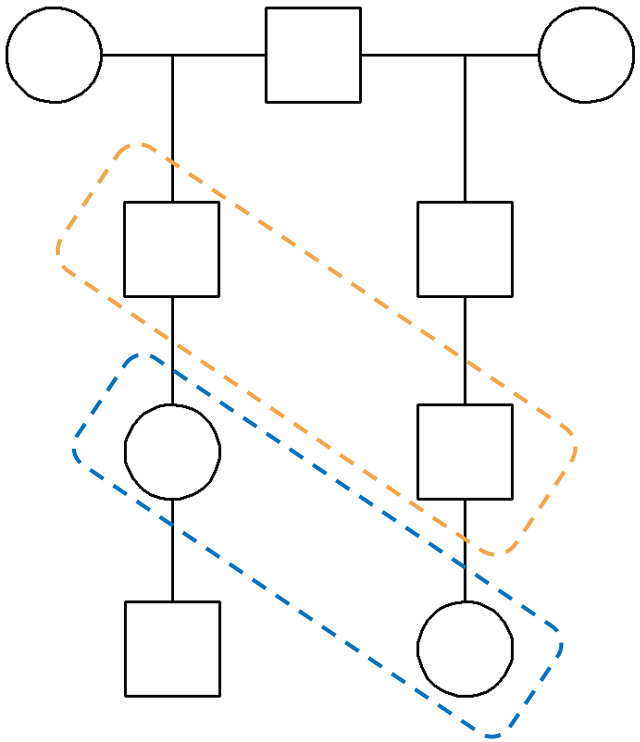


Half siblings

Half first cousins

Half second cousins

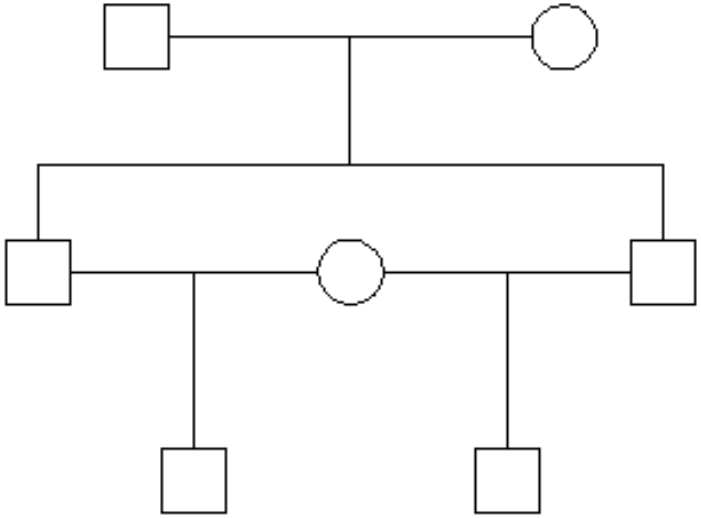
Half relationships



Half-uncle - half-nephew
(*half avuncular*)

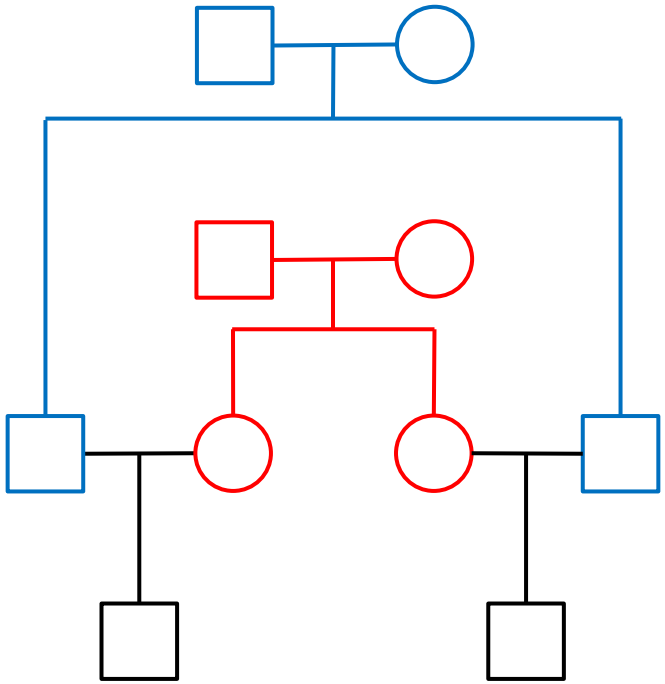
Half first cousins
once removed

More complicated relationships



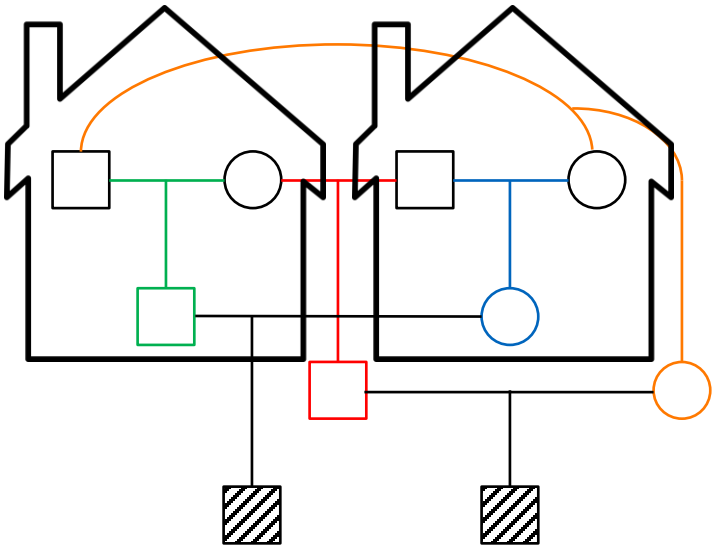
First cousins + half siblings = 3/4 siblings

Double relationships



Double first cousins

The connoisseur's favourite



Quadruple half first cousins!

What software exists to create pedigrees?

It depends!

- medical genetics
- forensic genetics
- animal pedigrees
- amateur genealogy

In this course:

- QuickPed
- R

https://magnusdv.shinyapps.io/quickped/

QuickPed: An Interactive Pedigree Creator

New app design!
Discover the new features
Or stay with the old version: QuickPed3

Purpose: QuickPed lets you rapidly create attractive pedigree plots, save them as images or text files, and analyse the relationships within them.

Instructions: Choose a suitable start pedigree and modify it by clicking on individuals and using appropriate buttons. For example, to add a male child, select the parent(s) and press the icon. Check out the online user manual for various tips and tricks, including an introduction to relatedness coefficients.

Citation: If you use QuickPed in a publication, please cite this paper: Vigeland MD (2022). QuickPed: an online tool for drawing pedigrees and analysing relatedness. *BMC Bioinformatics*, **23**. DOI:10.1186/s12859-022-04759-y.

Quick start

Built-in pedigree

Trio

or

Load a ped file

or

Random pedigree

Reset all

Modify

Add

Sex

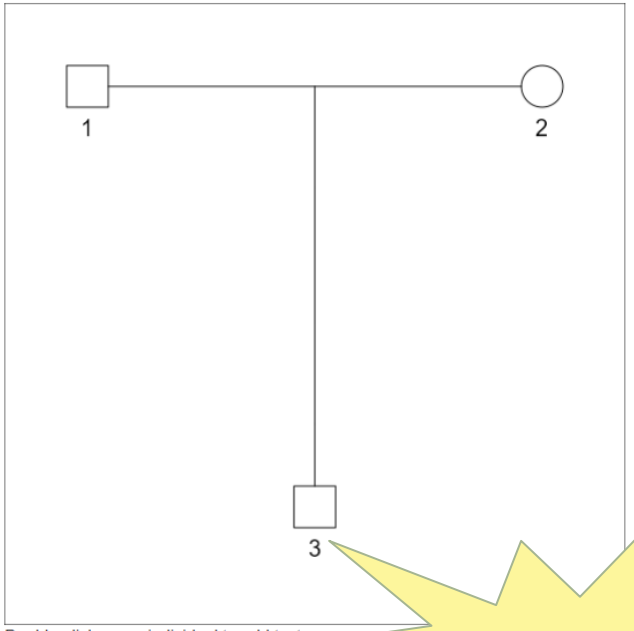
Style

Fill

Twins MZ / DZ

Remove

Undo



Double-click on an individual to add text

Labels

1, 2, 3, .. I-1, I-2, ..

Show all Hide all

1
2
3

Update

Plot settings

Width 430 **Height** 430

Cex 1,4 **Symbols** 1

Margins 3

Other options (beta)

- Straight legs
- Arrows

R code

PNG PDF

Relationships

Ped file

Include

- Headers
- Family ID
- Affection status

Save ped file

DEMO!

What software exists to create pedigrees?

It depends!

- medical genetics
- forensic genetics
- animal pedigrees
- amateur genealogy

In this course:

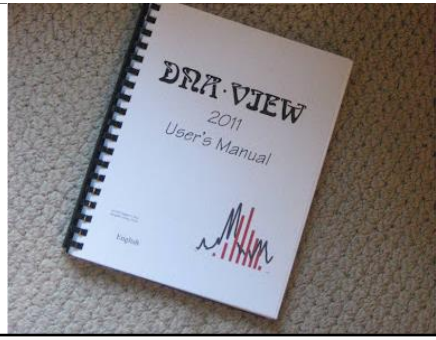
- QuickPed
- R

Crash course in R

What is R? (And why should you care?)

- A framework for statistical computing
  - calculator
  - data handling and numerical analysis
  - flexible plotting
  - programming language
  - external packages
    - anyone can make one
    - thousands!

Familias.no



BONAPARTE

Designed, built and proven for real world case work

Pros
- free!
- very widely used
- anything is possible (but not always easy)
- scripting --> reproducibility

Cons
- learning curve
- packages come and go

Oh boy, that sounds great!

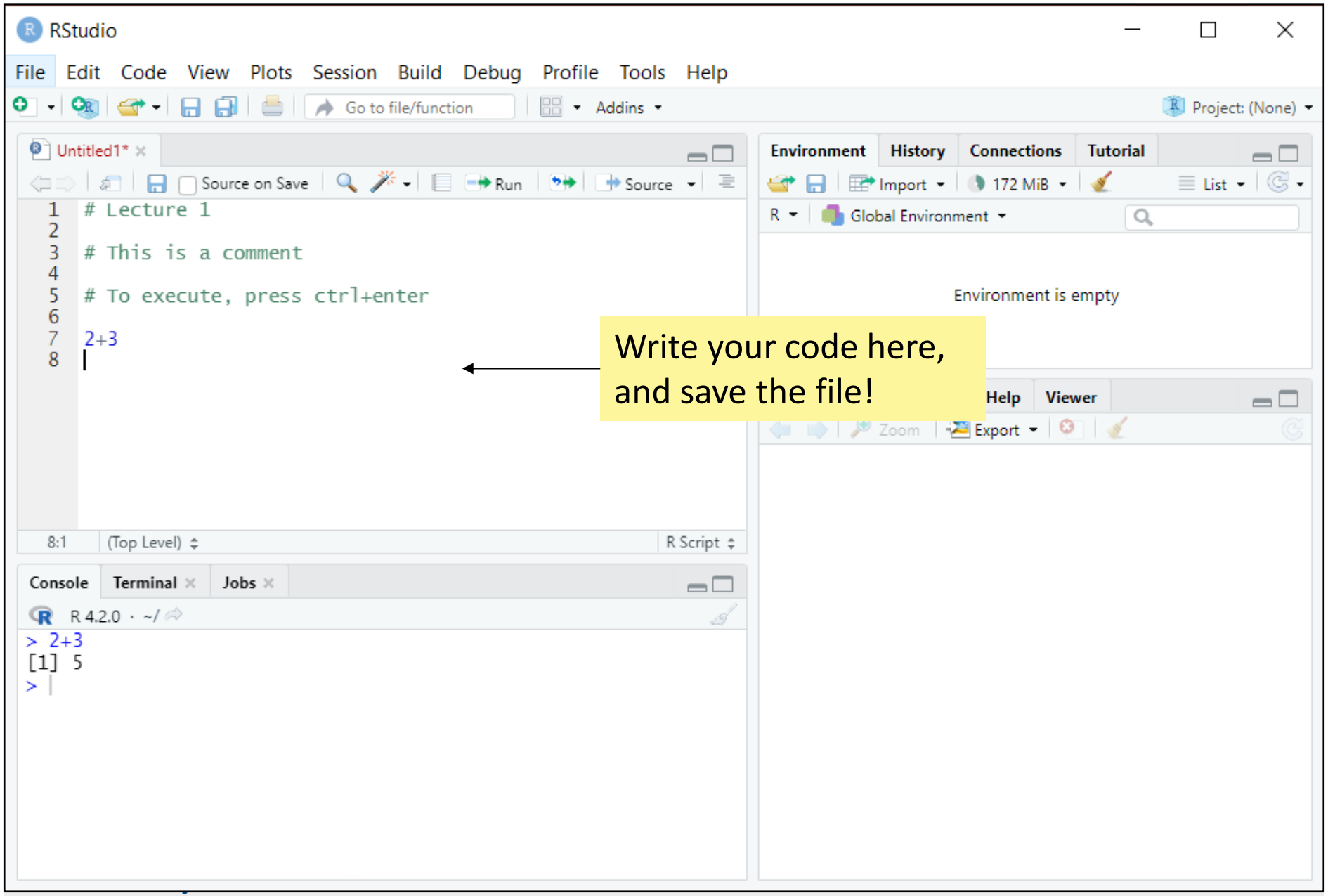
I which I knew R ...

Don't worry!

It's not that hard.

Here is a quick intro to R that contains most of what you need

RStudio


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
Lecture 1
This is a comment
To execute, press ctrl+enter
2+3
8:1
(Top Level)
R Script
Console
Terminal
Jobs
R 4.2.0 · ~/
> 2+3
[1] 5
>
Environment
History
Connections
Tutorial
Import
172 MiB
List
R
Global Environment
Environment is empty
Help
Viewer
Zoom
Export
Write your code here, and save the file!

Basic calculations

```
>   2 + 3
[1] 5
>   2+      3
[1] 5
>   (1 + 2) * 3
[1] 9
>   4^2
[1] 16
>   log(100)
[1] 4.60517
>   log(100, base = 10)
[1] 2
>   log10(100)
[1] 2
```

Spaces don't matter

$e^{4.60517} \approx 100$

Variables

I use this

Two (mostly synonymous) ways to assign values: **=** or **<-**

```
>   a = 5      or   a <- 5
>   b = 2      or   b <- 2
>   a
[1] 5
>   a - 2*b
[1] 1
```

Changing a variable:

```
>   a = a+1
>   a
[1] 6
```

Common beginners' mistake: forgetting to assign after change

Creating new variables from old:

```
>   newVar = a^b
>   newVar
[1] 36
```

Most programmers stick to either **camelCase** or **snake_case** when naming their variables

Vectors

```
>   c(3, 2, 6, -1)
[1]  3  2  6 -1
>   4:20
[1]  4  5  6  7  8  9 10 11 12
[10] 13 14 15 16 17 18 19 20
>   5:7 - 4
[1] 1  2  3
>   c(10,20,30,40) + c(1,3,8,0)
[1] 11 23 38 40
>   seq(from = 2, to = 15, by = 3)
[1]  2  5  8 11 14
```

The **c()** operator!

The **':'** operator
(shortcut for consecutive numbers)

There is a help page
for every function!
> **?seq**

Character vectors:

```
>   c("Alice", "Bob")
```

Logical vectors:

```
>   c(TRUE, FALSE, T, F)
[1] TRUE FALSE TRUE FALSE
```

Built-in logcial constants:
TRUE short form: **T**
FALSE short form: **F**

Matrix-like containers

Data frames: Collects vectors of the same length

```
>   x = data.frame(Name   = c("Ali", "Bob", "Joe"),
                   Weight = c(75, 81, 70))
>   x
  Name Weight
1  Ali     75
2  Bob     81
3  Joe     70
```

Use **$** to refer to columns: **x$Name**

Matrices:

```
>   x = matrix(1:12, nrow = 3, ncol = 4)
>   x
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Note: No **$** for matrices!

First column: **x[, 1]**
First row: **x[1,]**

Faster, but less flexible. Good for all-numeric (or all-character) data

Lists

```
>   a = list(good = 1:3, bad = 0)
>   a
$good
[1] 1  2  3

$bad
[1] 0
>   a$good
[1] 1  2  3
```

Alternative to **$**:
a[["good"]]

Easy to change lists:

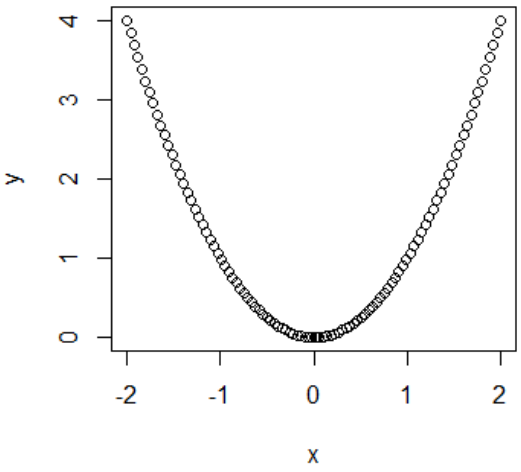
```
>   a$bad = NULL              (delete item)
>   a$ok = -1                 (add new item)
>   a$good = c(a$good, 10)    (modify item)
>   a
$good
[1] 1  2  3  10

$ok
[1] -1
```

Basic plotting

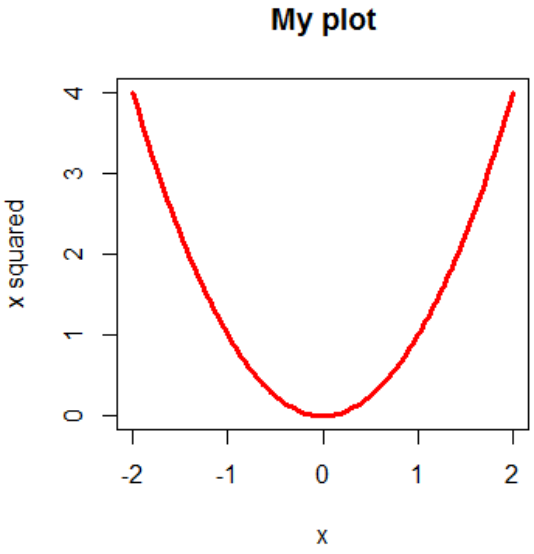
Let's plot the graph of $y = x^2$!

```
>   x = seq(-2, 2, length = 100)
>   y = x^2
>   plot(x, y)
```



Many options to play with...

```
>   plot(x, y, type="l", lwd = 3, col = "red",
         ylab = "x squared", main = "My plot!")
```



The pipe: |>

Introduced in R version 4.1

- Enables function chaining. Often easier to read.

Consider this code:

```
>  a = exp(2)
>  b = log(a, base = 10)
>  rep(b, times = 3)
[1] 0.868589 0.868589 0.868589
```

One-liner producing the same:

```
>  rep(log(exp(2), base = 10), times = 3)
[1] 0.868589 0.868589 0.868589
```

With piping:

```
>  exp(2) |> log(base = 10) |> rep(times = 3)
[1] 0.868589 0.868589 0.868589
```

Purists: Line break after each pipe

```
exp(2) |>
  log(base = 10) |>
  rep(times = 3)
```

R stuff skipped in this brief introduction

- User-defined functions
- Loops, `apply()`, `lapply()`, etc.
- Basic statistics, linear models + +
- Random numbers
- The "tidyverse" for data science
- ... and LOTS of other things...

Installing packages

To access the functions of an external package, you must:

- install the package
  - downloads it to your computer
  - this is done only once
  - `install.packages()`
- load it into R
  - every new session
  - `library()`

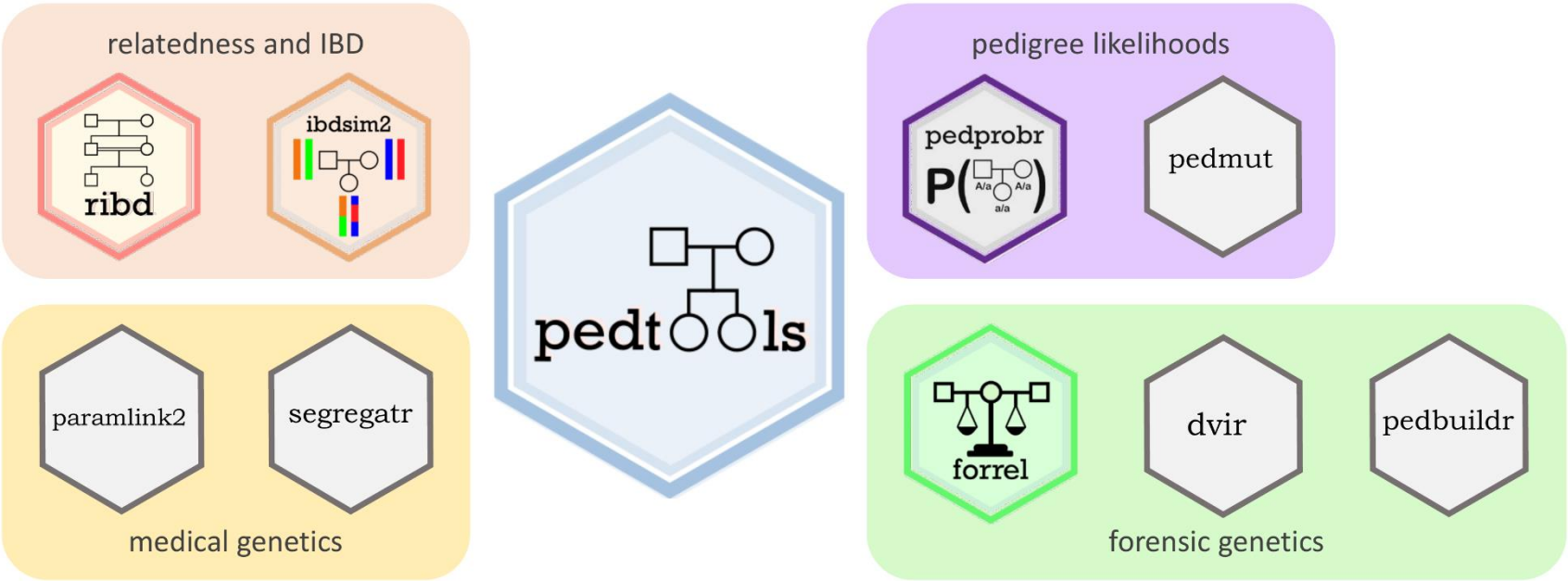
To check if a package is installed, simply try to load it:

```
> library(pedsuite)
```

If you get an error, do:

```
> install.packages("pedsuite")
```

The pedsuite: A collection of packages for pedigree analysis in R



Home page:
https://magnusdv.github.io/pedsuite

Source code available on GitHub:
https://github.com/magnusdv

The pedsuite: A collection of packages for pedigree analysis in R

relatedness and IBD

ribd

ibdsim2

likelihoods

pedmut

paramlink2

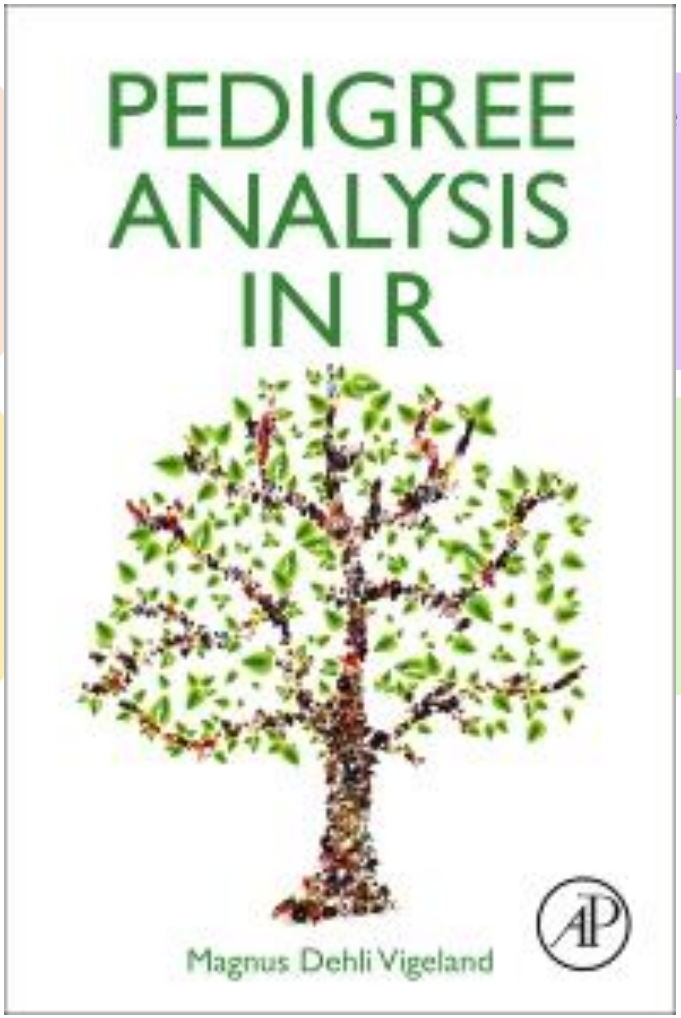
segregatr

medical genetics

dvir

pedbuildr

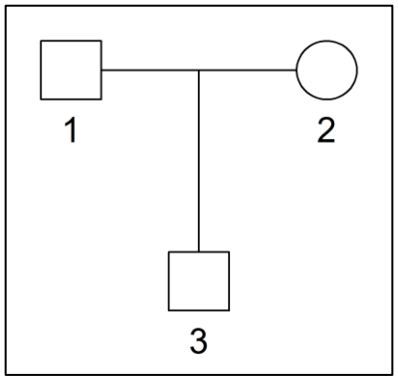
forensic genetics



Building pedigrees

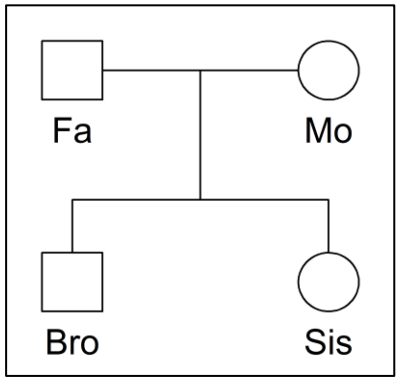
```
>   library(pedsuite)

>   x = nuclearPed()
>   plot(x)
```



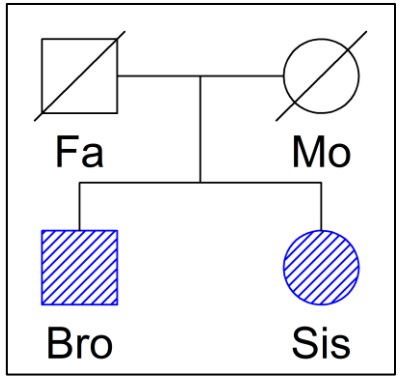
Names and sex:

```
>   y = nuclearPed(father = "Fa",
                   mother = "Mo",
                   child = c("Bro", "Sis"),
                   sex = 1:2)
>   plot(y)
```



Many ways to tweak the plot!

```
>   plot(y,
         deceased = c("Fa", "Mo"),
         hatched = c("Bro", "Sis"),
         col = list(blue = c("Bro", "Sis")),
         cex = 1.5)
```



Some useful functions

Create: basic

- singleton
- nuclearPed
- linearPed
- halfSibPed
- avuncularPed
- cousinPed

Create: complex

- ancestralPed
- doubleCousins
- quadHalfFirstCousins
- fullSibMating
- randomPed

Manipulate

- addSon
- addDaugher
- addParents
- addChildren

- swapSex
- relabel
- removeIndividuals

- branch
- subset

- mergePed
- breakLoops

Member subsets

- founders
- nonfounders
- leaves
- males
- females
- typedMembers
- untypedMembers

Relatives

- father
- mother
- children
- siblings
- grandparents
- spouses

- ancestors
- descendants
- unrelated

Another example

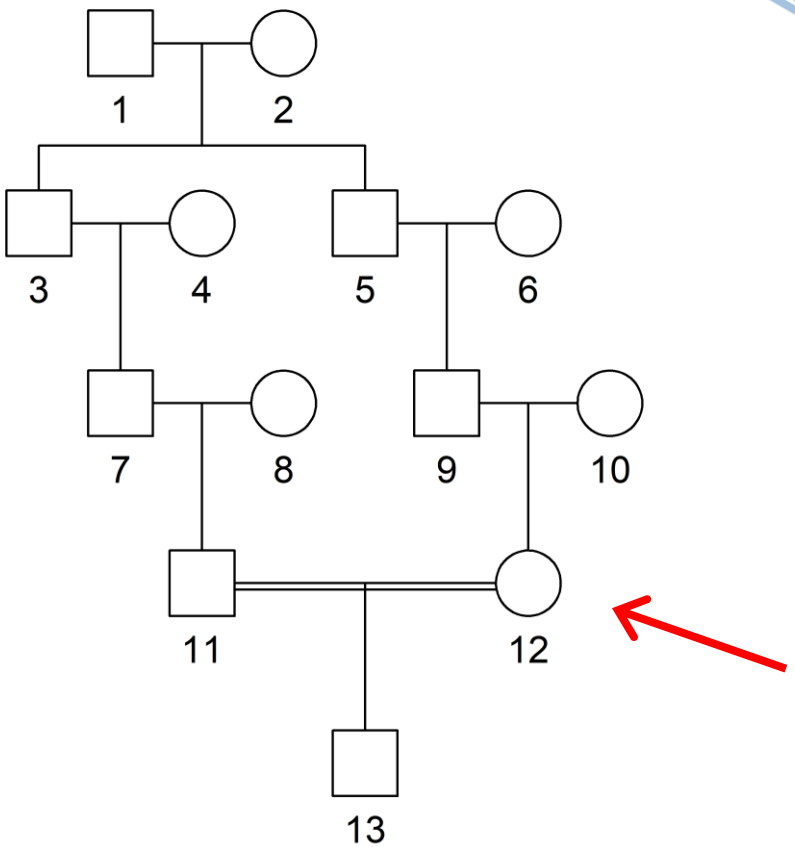
```
>   x = cousinPed(2)
>   plot(x)
```

Change gender:

```
>   x = swapSex(x, 12)
>   plot(x)
```

Add inbred child

```
>   x = addSon(x, parents = 11:12)
>   plot(x)
```



Remember or pipe!

- Store the result after each change!
- It is OK to use the same name
  (if you don't need the previous object)

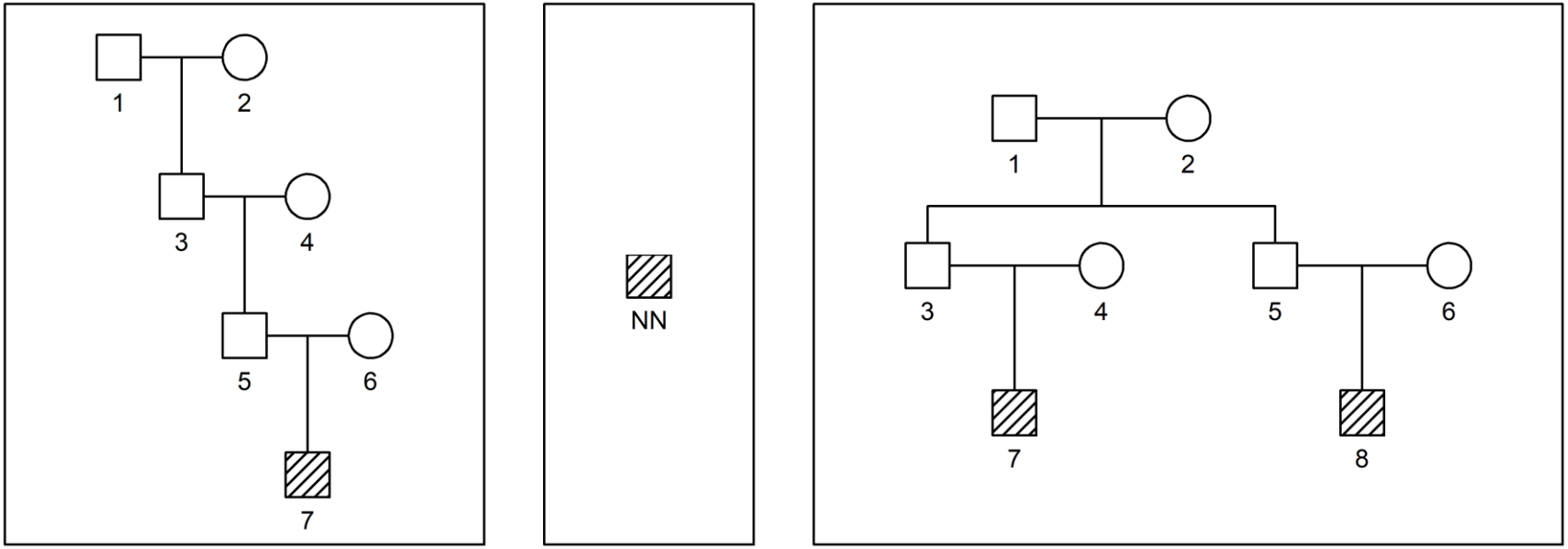
Shortcut command for this pedigree

```
> x = cousinPed(2, child = TRUE)
```

List of pedigrees

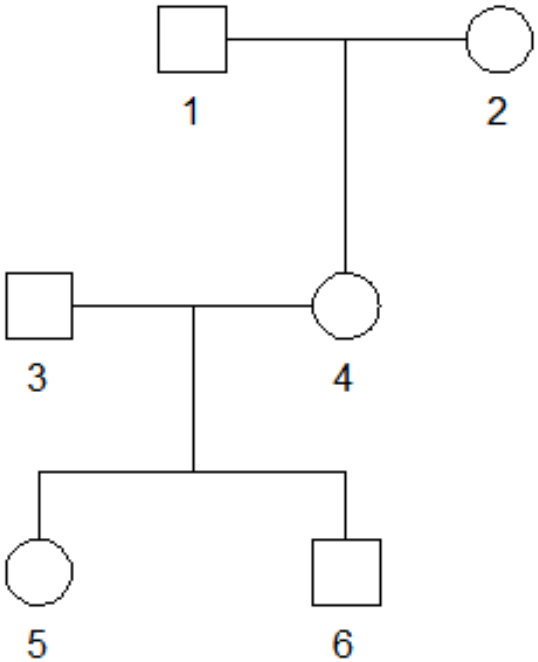
```
>   peds = list(linearPed(3),
                singleton("NN"),
                cousinPed(1))

>   plotPedList(peds,
                widths = c(2, 1, 3),
                hatched = leaves)
```



Alternative pedigree creation: .ped file

A text file describing a pedigree structure.



0 if founder

famid	id	fid	mid	sex
1	1	0	0	1
1	2	0	0	2
1	3	0	0	1
1	4	1	2	2
1	5	3	4	2
1	6	3	4	1

Contents of *example.ped*

In pedtools:

```
> x = readPed("example.ped")
> plot(x)
```

Columns

famid = family ID (optional)
id = individual ID
fid = ID of father
mid = ID of mother
sex = 1 (male), 2 (female) or 0 (unknown)

Oh my! Do I have to write these ped-files by hand?

No, that is tedious and error-prone!

Better: **QuickPed**

Now: Exercises!