

ISFG Summer School 2025 - Workshops 10.1 and 10.2

# **Kinship Statistics and Pedigree Analysis**

**Teachers**

Magnus Dehli Vigeland

Thore Egeland

Home page:

[https://magnusdv.github.io/pedsuite/articles/web\\_only/course-isfg2025.html](https://magnusdv.github.io/pedsuite/articles/web_only/course-isfg2025.html)

## Schedule

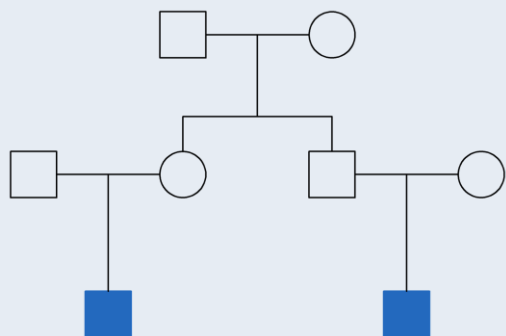
The course runs from 08 to 12 (CEST) each day, with a 15 minute break in the middle. The following schedule is tentative:

### Sept 4 (Thu). Pedigree analysis: Basic

- 08:00–09:00 **Pedigrees and measures of relatedness** (MDV)
- 09:00–10:00 Exercises I
- 10:00–10:15 *Break*
- 10:15–11:00 **Kinship testing** (TE)
- 11:00–11:45 Exercises II
- 11:45–12:00 Summary and discussion

### Sept 5 (Fri). Pedigree analysis: Advanced

- 08:00–09:00 **Relatedness inference and pedigree reconstruction** (MDV)
- 09:00–10:00 Exercises III
- 10:00–10:15 *Break*
- 10:00–11:00 **Disaster victim identification** (TE)
- 11:00–11:45 Exercises IV
- 11:45–12:00 Summary and discussion



# Lecture 1: Pedigrees and measures of relatedness

ISFG Summer School 2025 - Workshop 10.1

**Kinship Statistics and Pedigree Analysis: Basic**

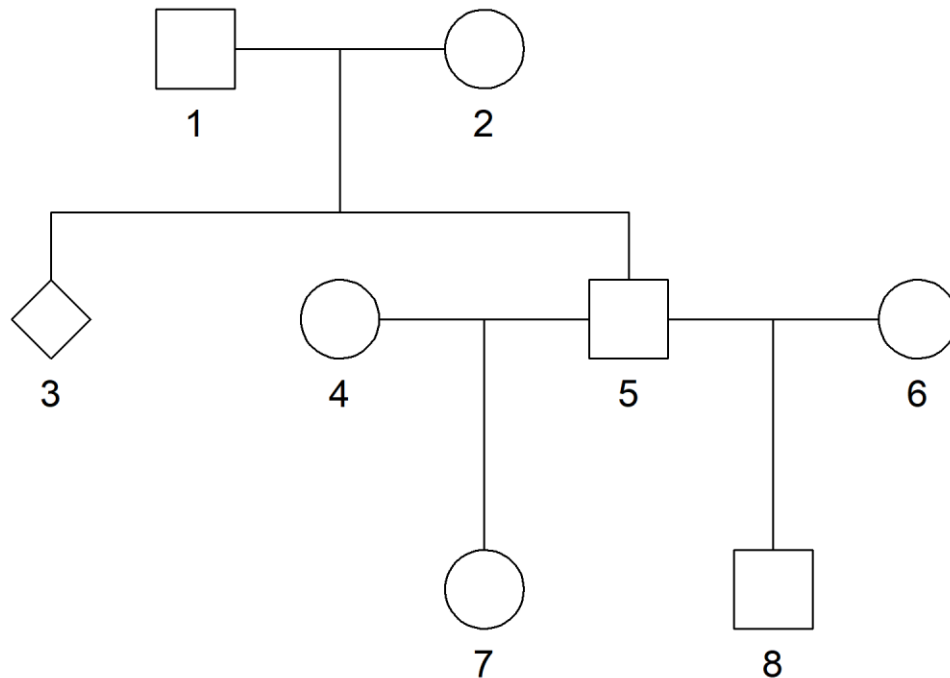
Magnus Dehli Vigeland

# Outline

- Part I: *Pedigrees*
  - Conventions and terminology
  - QuickPed
- Part II: *Measures of relatedness*
  - Identity by descent (IBD)
  - Kinship/inbreeding coefficients
  - IBD triangle
  - Realised relatedness
- (Part III: Crash course in R)

# Part I: Pedigrees

# Conventions and terminology

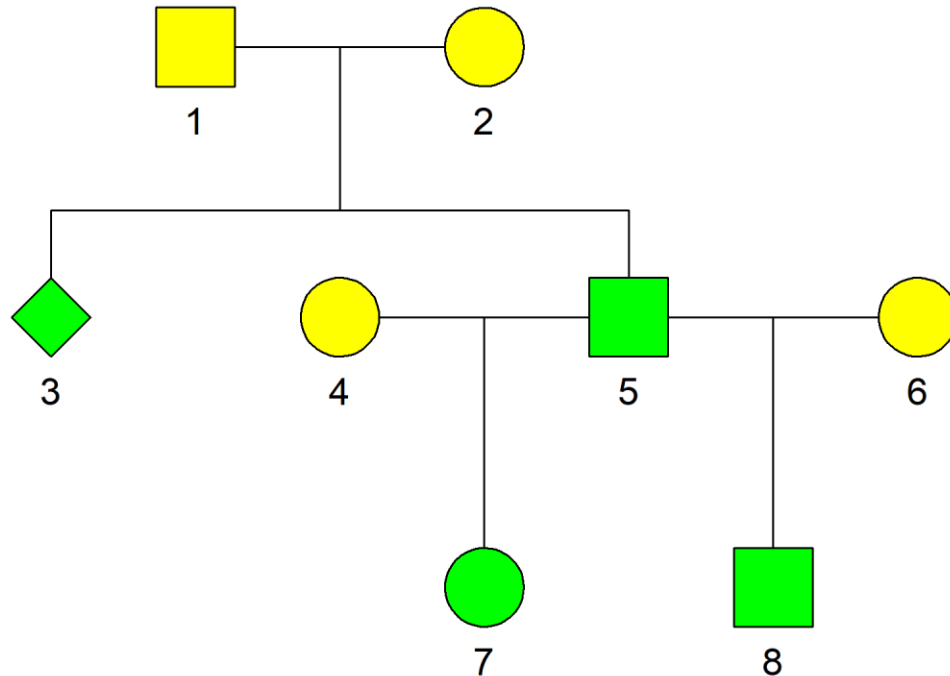


□ = male

○ = female

◇ = unknown

# Conventions and terminology



□ = male

○ = female

◇ = unknown

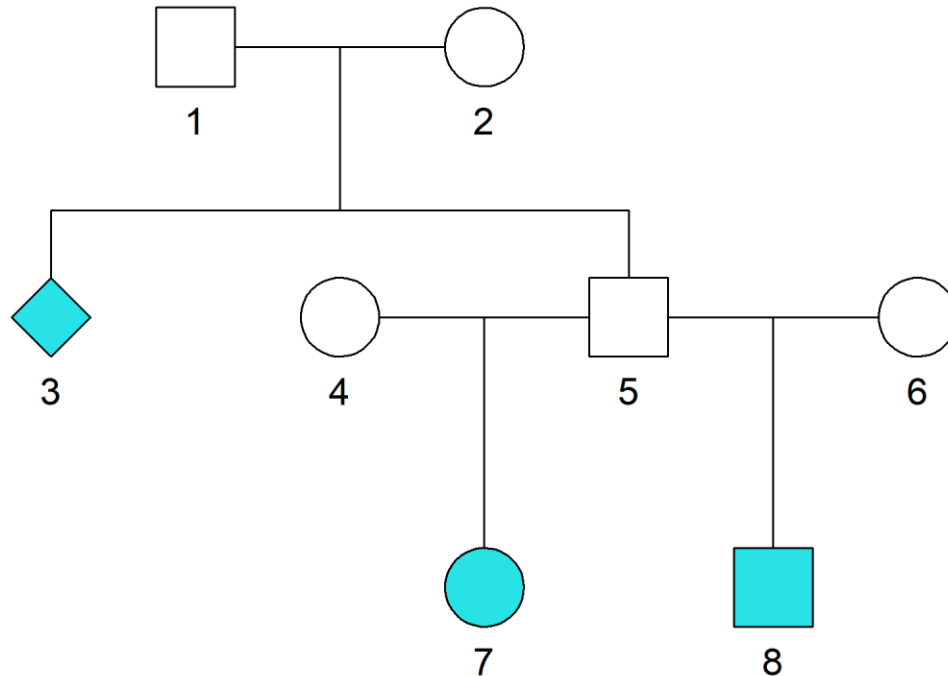
## **Founders**

No parents included  
in the pedigree

## **Nonfounders**

Parents are included

# Conventions and terminology



□ = male

○ = female

◇ = unknown

## **Founders**

No parents included  
in the pedigree

## **Nonfounders**

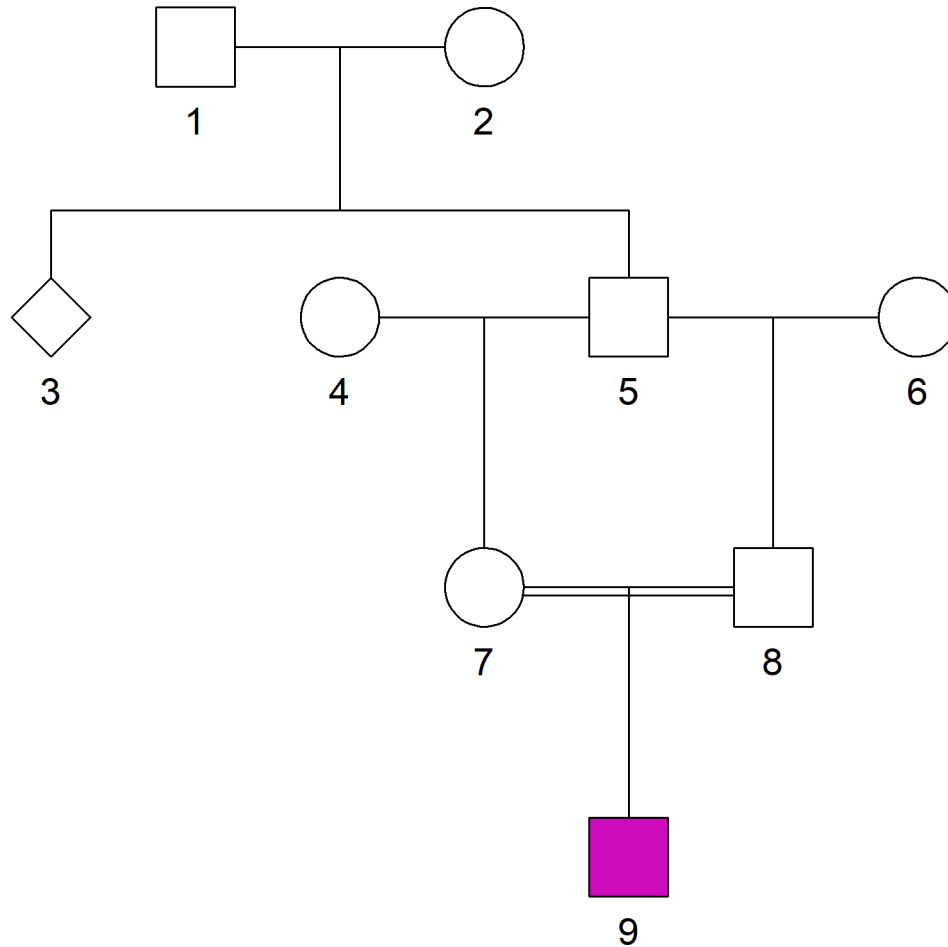
Parents are included

## **Leaves**

No children included



# Conventions and terminology



□ = male

○ = female

◇ = unknown

## **Founders**

No parents included  
in the pedigree

## **Nonfounders**

Parents are included

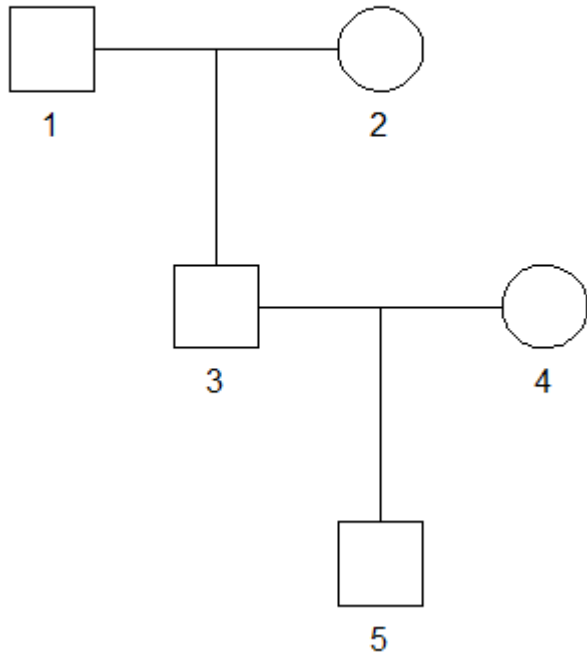
## **Leaves**

No children included

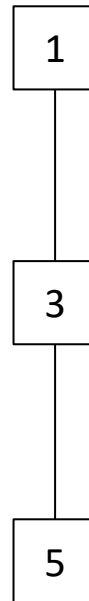
## **Inbred**

Children of related  
parents

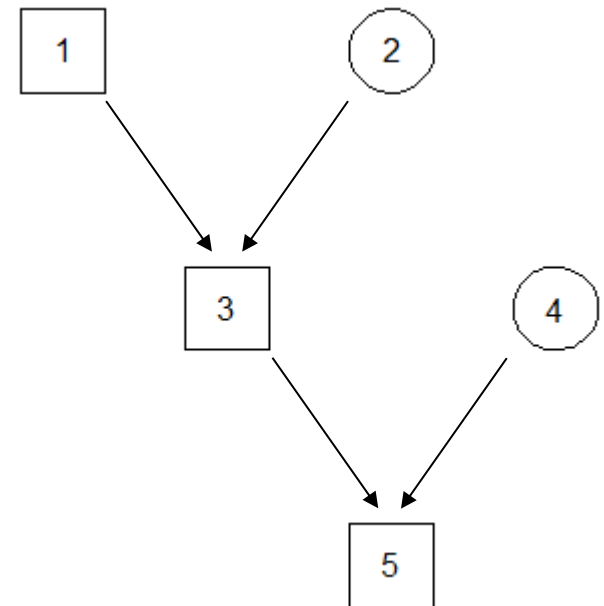
# Alternative ways of drawing pedigrees



Standard



Simplified

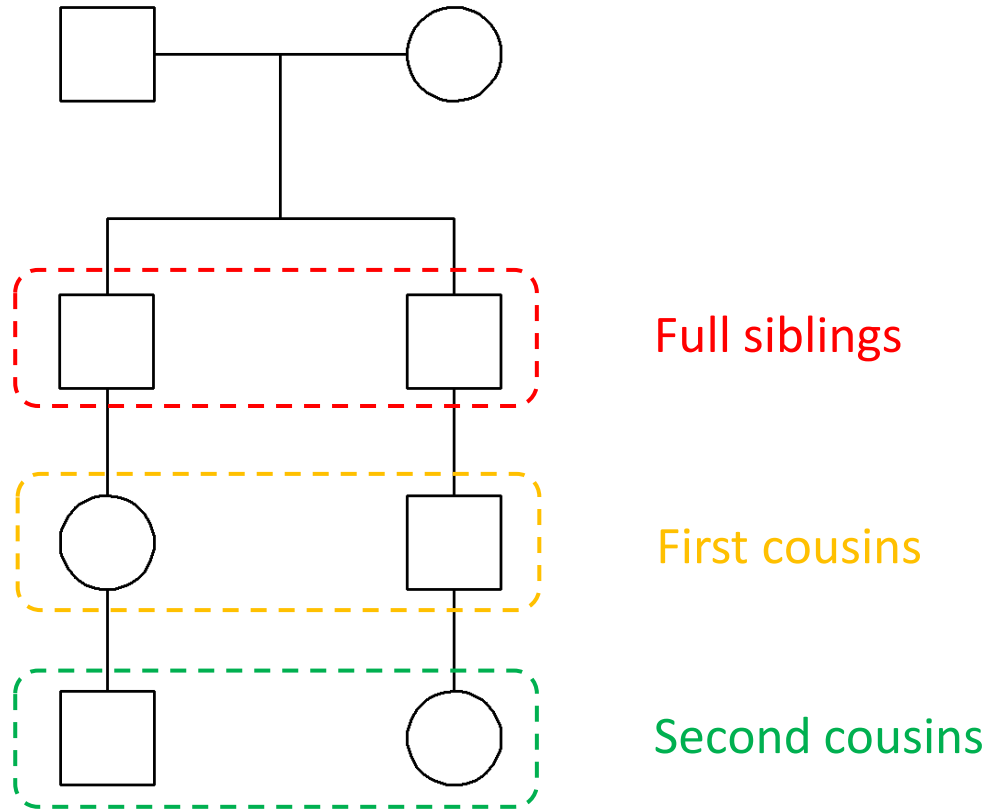


Directed acyclic graph

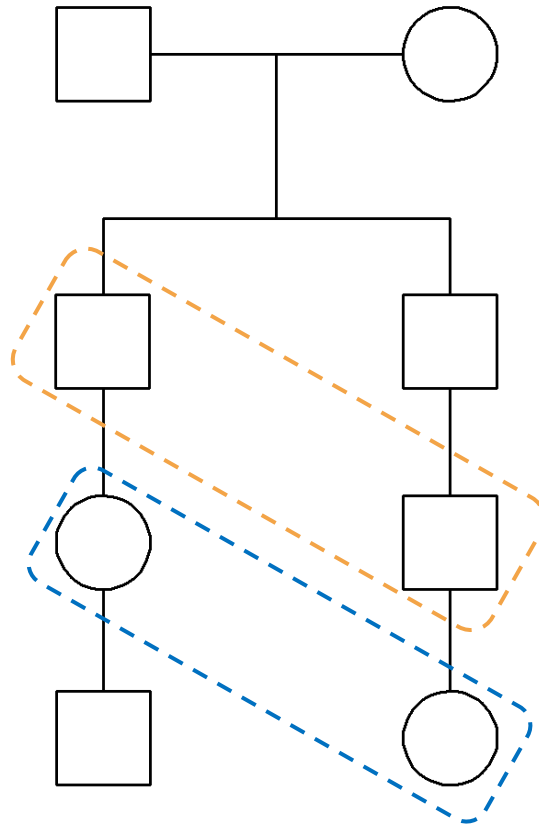
# Some common relationships

(and some less common...)

# Cousin relationships



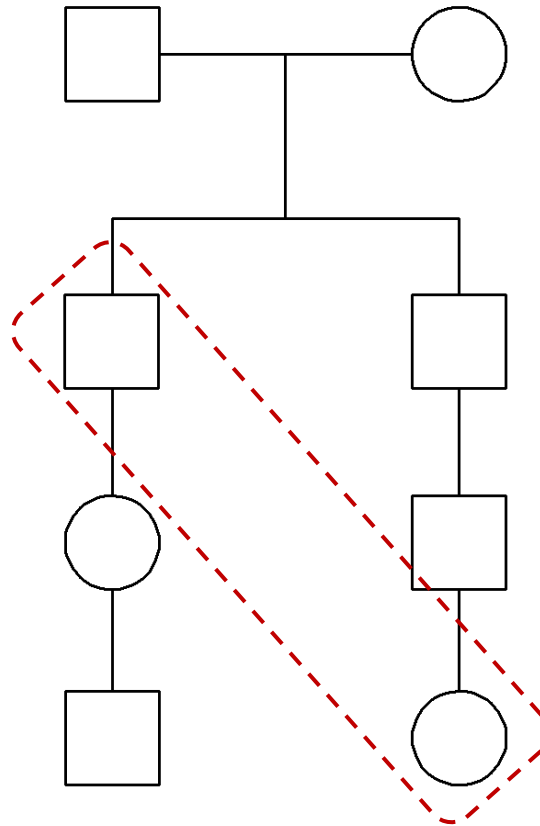
# Cousin relationships



Uncle - nephew  
(*avuncular*)

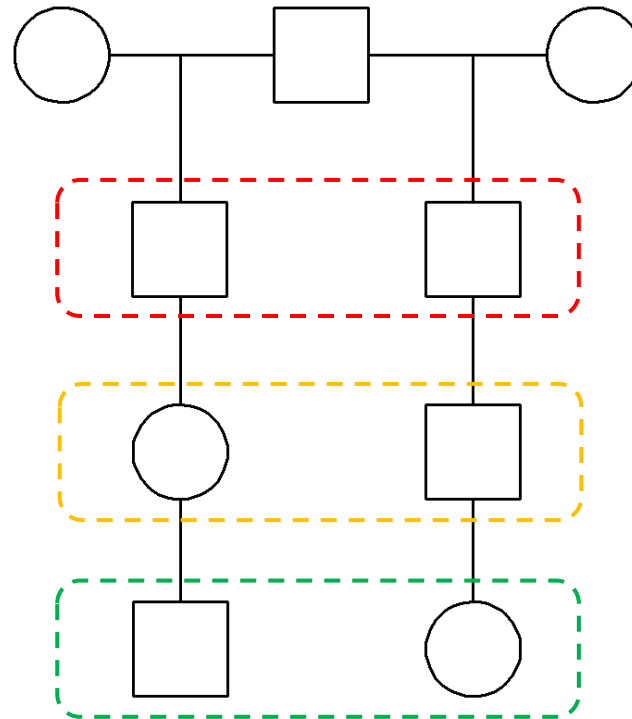
First cousins  
once removed

# Cousin relationships



Grand-uncle  
(or *great uncle*)

# Half cousin relationships

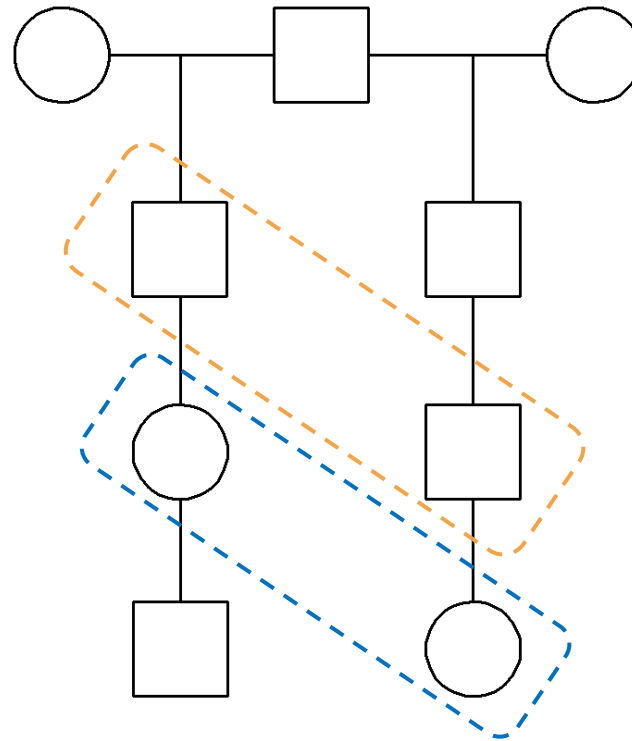


Half siblings

Half first cousins

Half second cousins

# Half cousin relationships

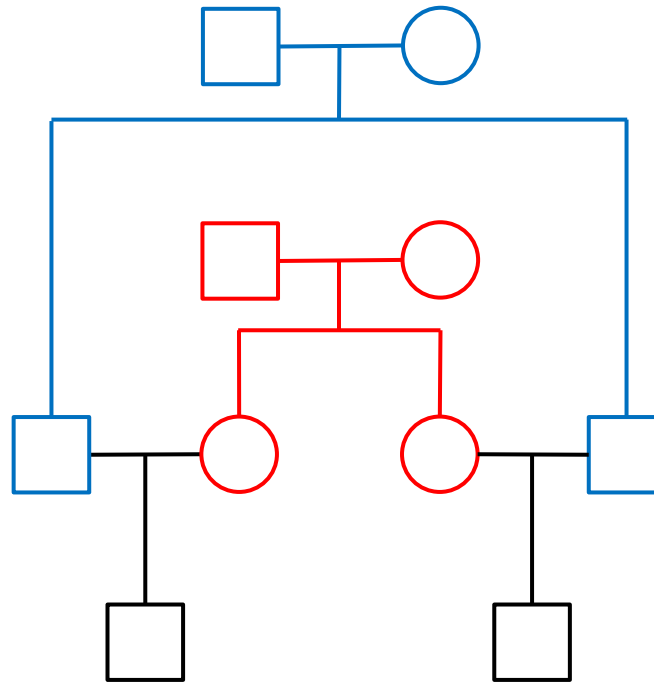


Half-uncle - half-nephew  
(*half avuncular*)

Half first cousins  
once removed

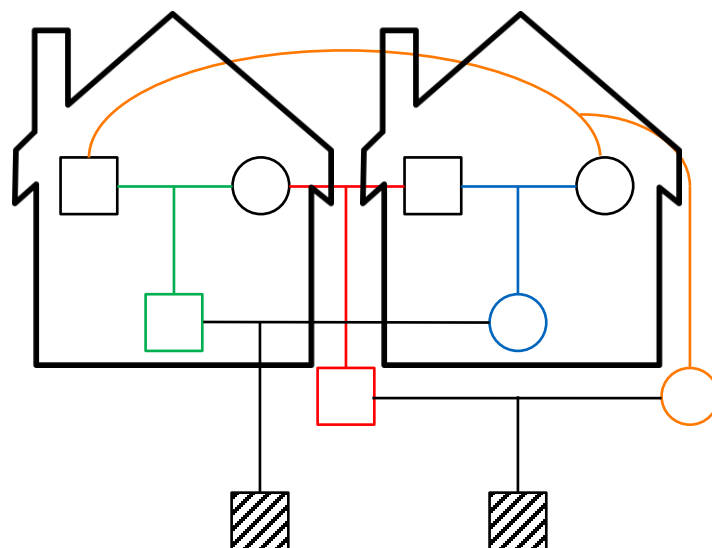


## Double relationships

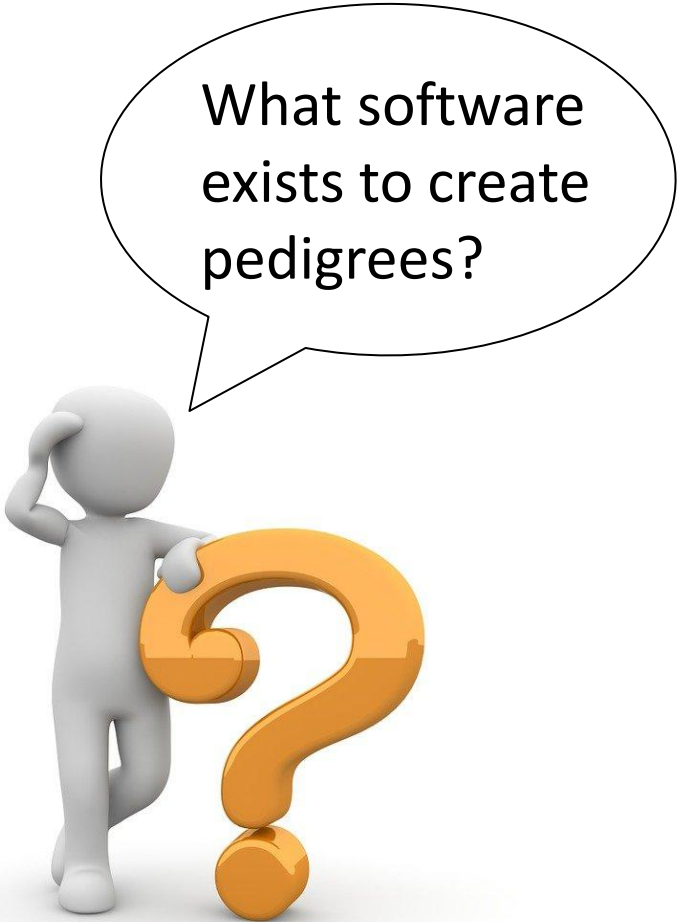


## Double first cousins

## 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




# 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](https://doi.org/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



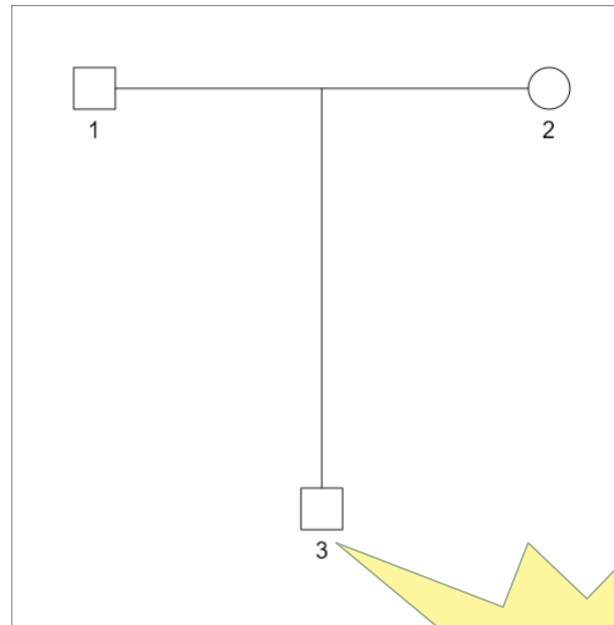
### Twins

MZ / DZ

### Remove



Undo



Double-click on an individual to add text

**DEMO!**

## 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

## Ped file

Include

☒ Headers

☐ Family ID

☐ Affection status

Save ped file

## Relationships



## Part II: Measures of relatedness

## Typical responses

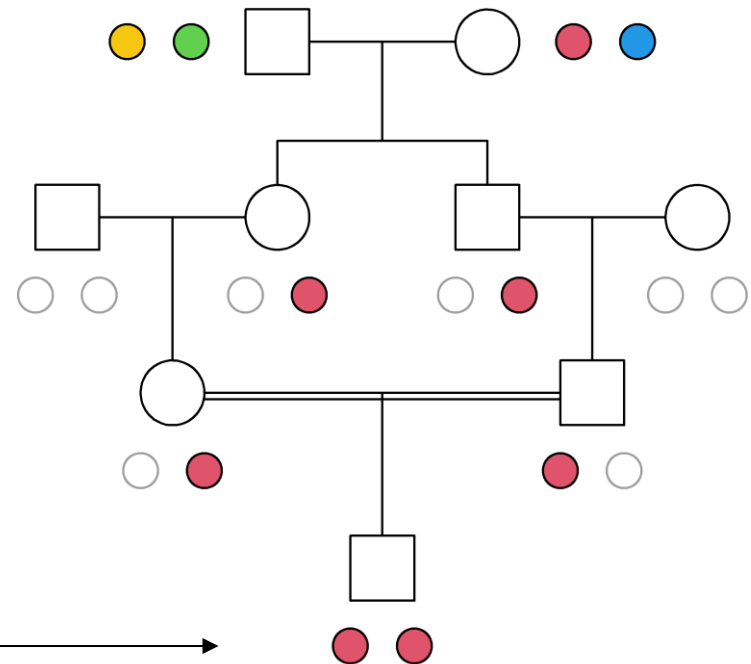
- being connected by family
- having a common ancestor...  
(not too far back)
- sharing DNA ...  
(more than unrelated people)

To make this precise, we need  
some terminology!



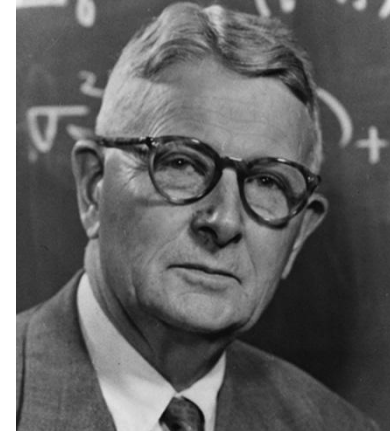
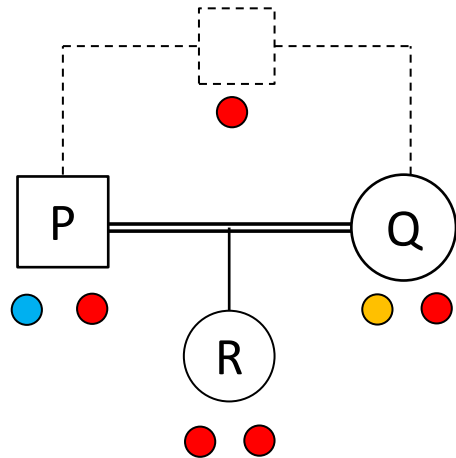
# IBD and autozygosity

- identity by descent (IBD)
  - when alleles have a common origin in the given pedigree
- autozygous
  - homozygous; alleles are IBD



Inbreeding coefficient  
 $f = Pr(\text{autozygosity})$

# Coefficient of kinship/inbreeding



Sewall Wright  
(1889 - 1988)

- Wright (1921): The kinship coefficient  $\varphi$

$$\varphi_{P,Q} = \Pr(\text{P and Q emit IBD alleles})$$

$$= \Pr(\text{R is autozygous})$$

$$= f_R$$

Kinship of parents = inbreeding of child

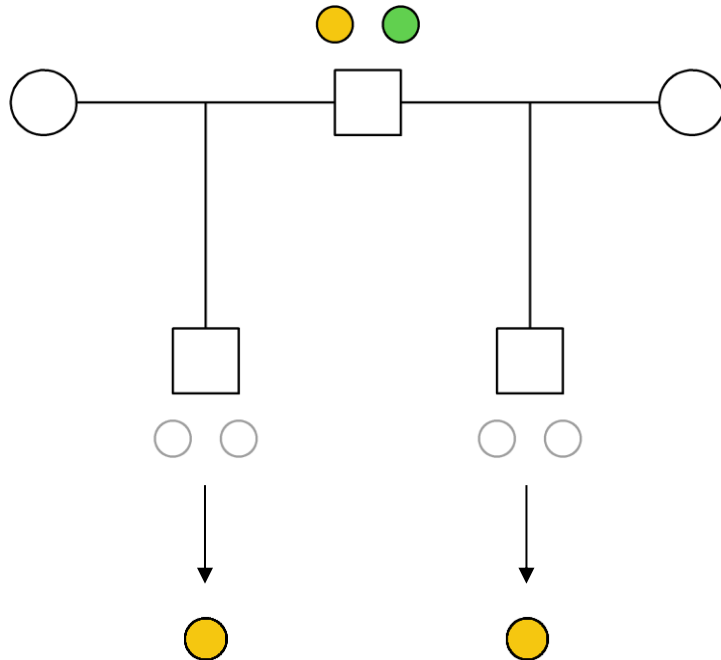
P and Q related



$$\varphi_{P,Q} > 0$$



## Example: Kinship coefficient of half siblings



### Kinship coefficient

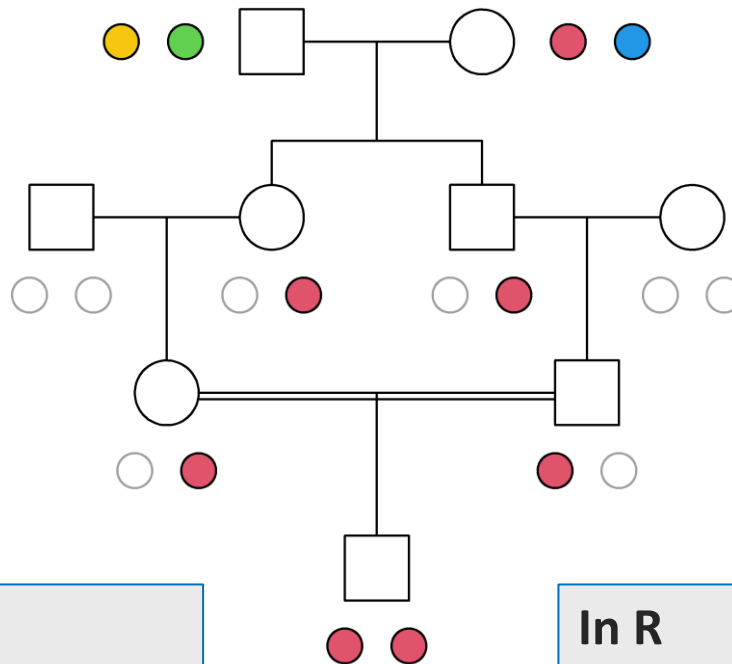
$$\begin{aligned}\varphi &= P(\text{yellow circle from both}) \cdot 2 \\ &= 0.5^4 \cdot 2 \\ &= 1/8\end{aligned}$$

↑  
green

# Inbreeding coefficient: Example

Wright's path formula:

$$\varphi_{P,Q} = \sum_A \sum_v \left(\frac{1}{2}\right)^{|v|+1} (1 + f_A)$$



By hand

$$\begin{aligned} f &= P(\text{red autozygous}) \cdot 4 \\ &= 0.5^6 \cdot 4 \\ &= 1/16 \end{aligned}$$

↑  
other  
colors

In R

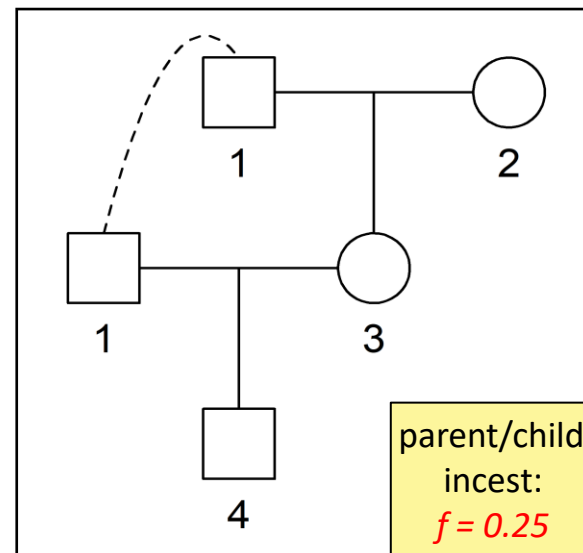
```
> library(pedsuite)

> x = cousinPed(1, child = T)
> inbreeding(x, ids = 9)
[1] 0.0625
```

# More kinship & inbreeding coefficients

Relationship	kinship $\varphi$ = $f$ of child
Parent-child	1/4
Full siblings	1/4
Half siblings	1/8
Grandparent-grandchild	1/8
Avuncular (uncle/aunt)	1/8
1st cousins	1/16
2nd cousins	1/64
3rd cousins	1/256

**Challenge**  
Different relationships  
with the same kinship!

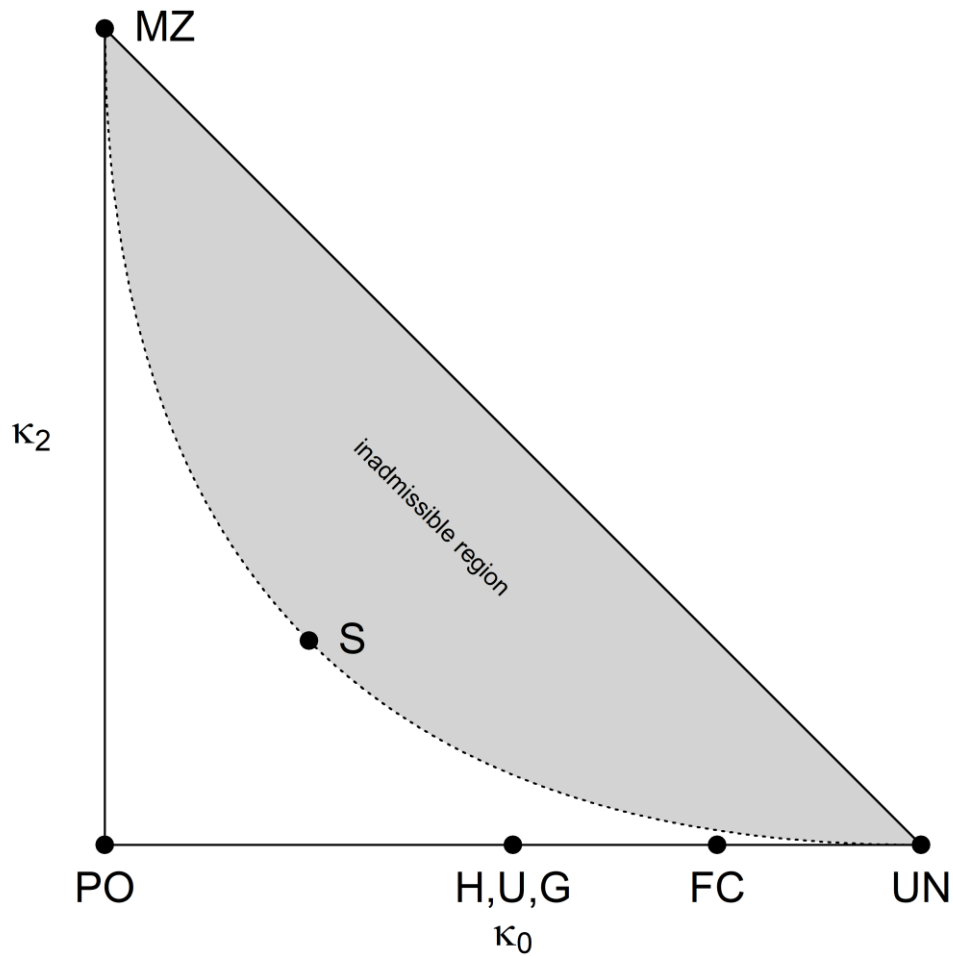


```
> x = nuclearPed(1, sex = 2) |>
  addSon(parents = c(1, 3))

> kinship(x, ids = c(1, 3))
[1] 0.25

> inbreeding(x, id = 4)
[1] 0.25
```

# The IBD triangle



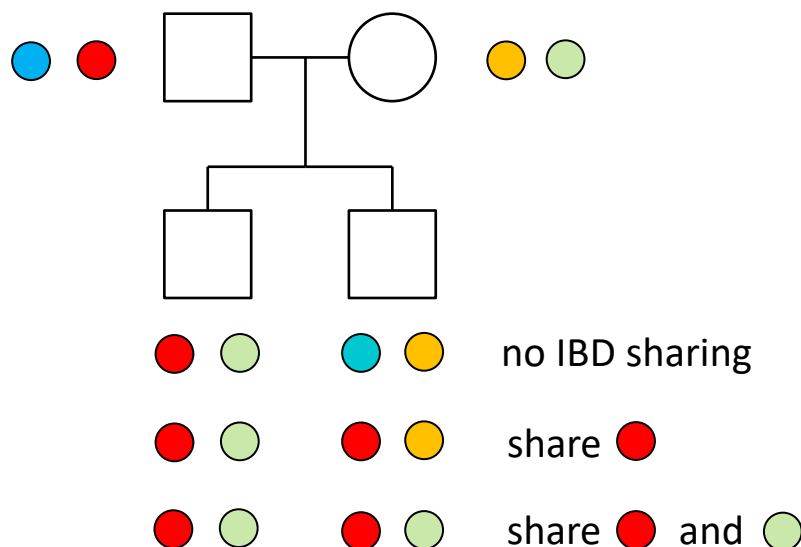
Charles Cotterman  
(1914-1989)



Elisabeth Thompson  
(1949 - )

# IBD coefficients

- Summary so far:
  - Two individuals are related if they can have IBD alleles
  - Their kinship coefficient measures the amount of IBD sharing
- Natural generalisation:
  - How *many* alleles are IBD in each locus?

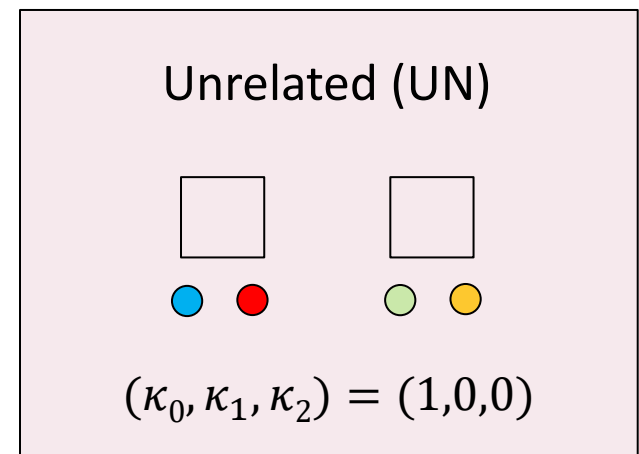
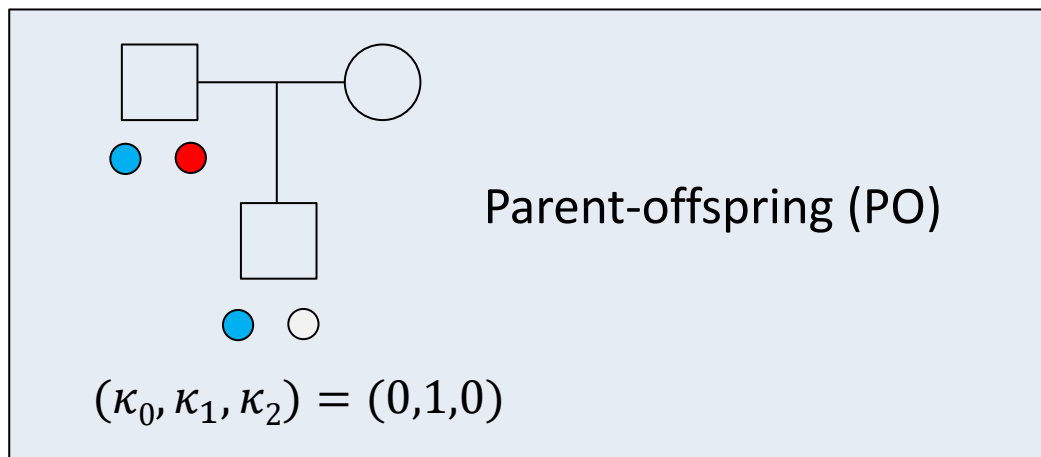
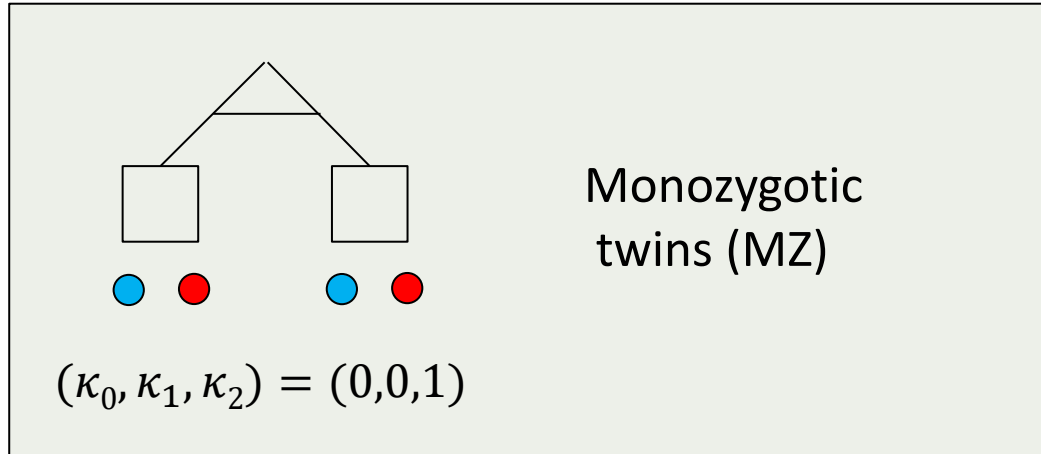


## Definition

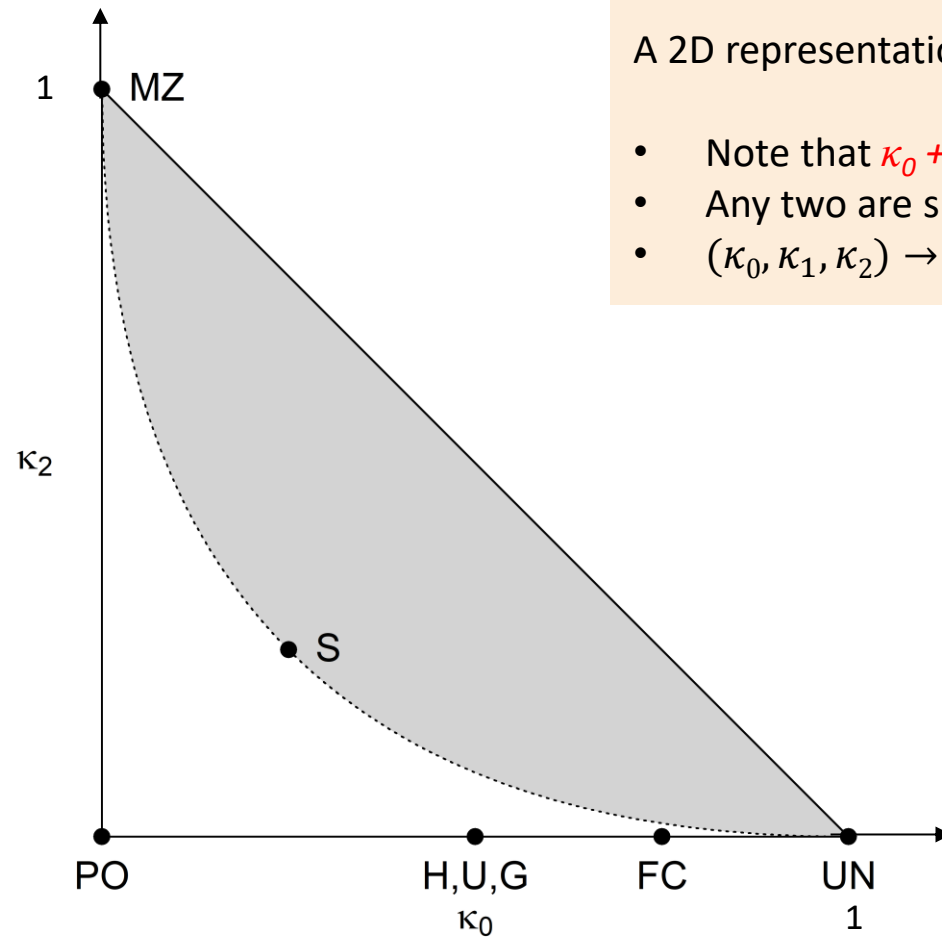
- $\kappa_0 = Pr(0 \text{ alleles IBD})$
- $\kappa_1 = Pr(1 \text{ alleles IBD})$
- $\kappa_2 = Pr(2 \text{ alleles IBD})$

(at random autosomal locus)

# Three trivial relationships



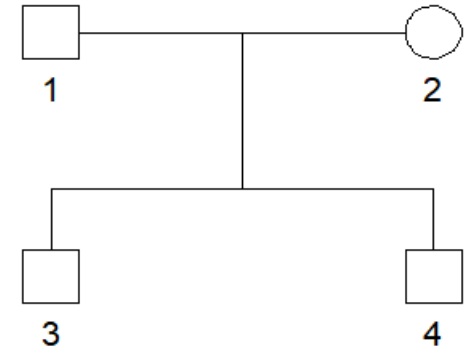
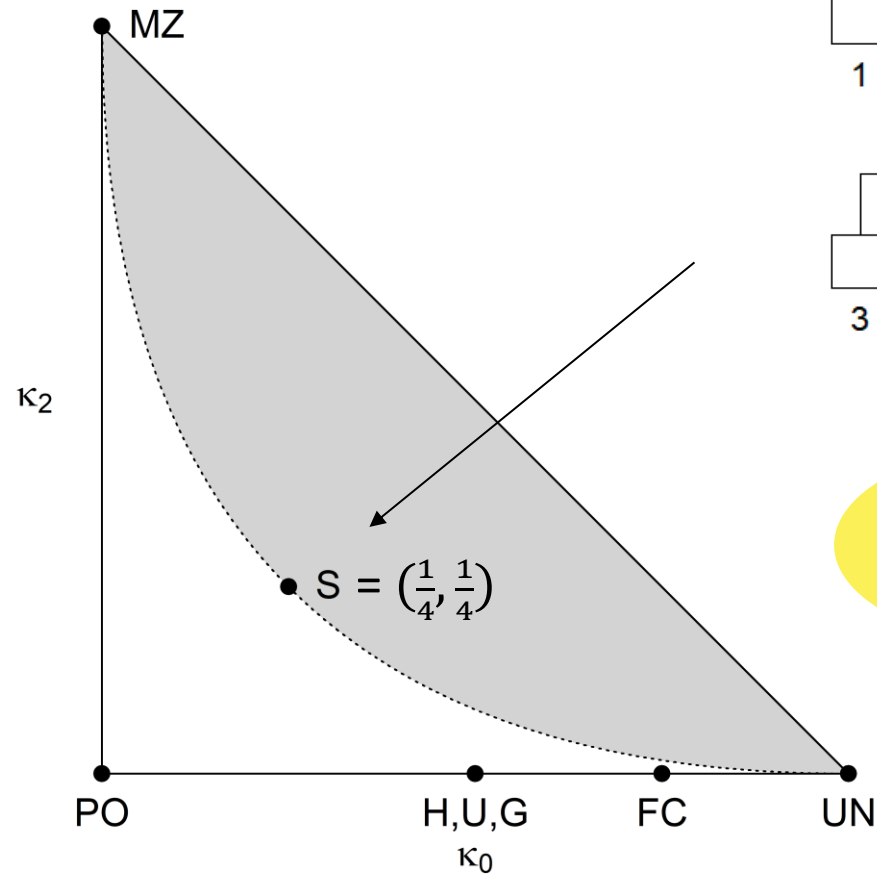
# The relatedness triangle



A 2D representation of  $(\kappa_0, \kappa_1, \kappa_2)$

- Note that  $\kappa_0 + \kappa_1 + \kappa_2 = 1$
- Any two are sufficient
- $(\kappa_0, \kappa_1, \kappa_2) \rightarrow (\kappa_0, \kappa_2)$

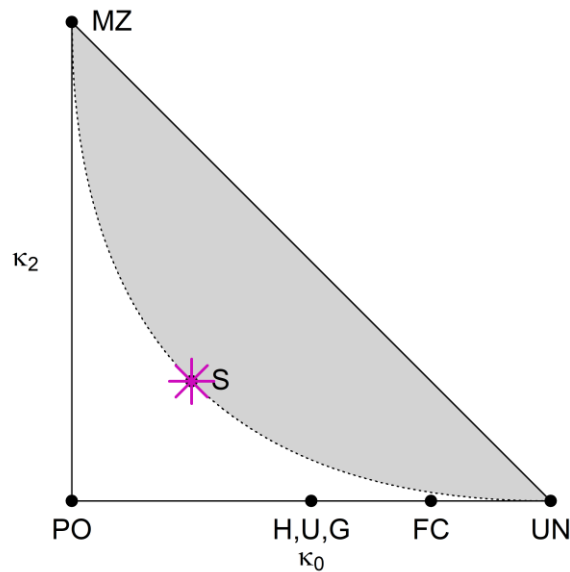
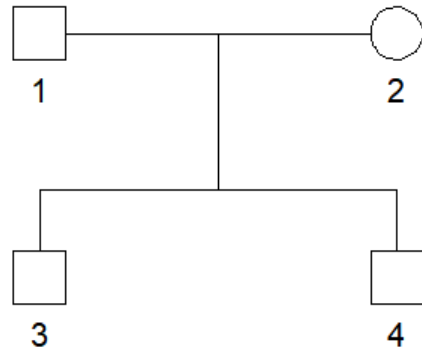
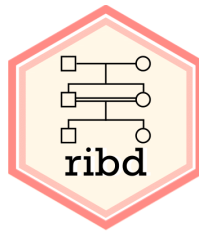
# What are the coefficients of full sibs?



Let's do this  
in R!



# ribd: Pedigree-based relatedness coefficients



```
> library(pedsuite)

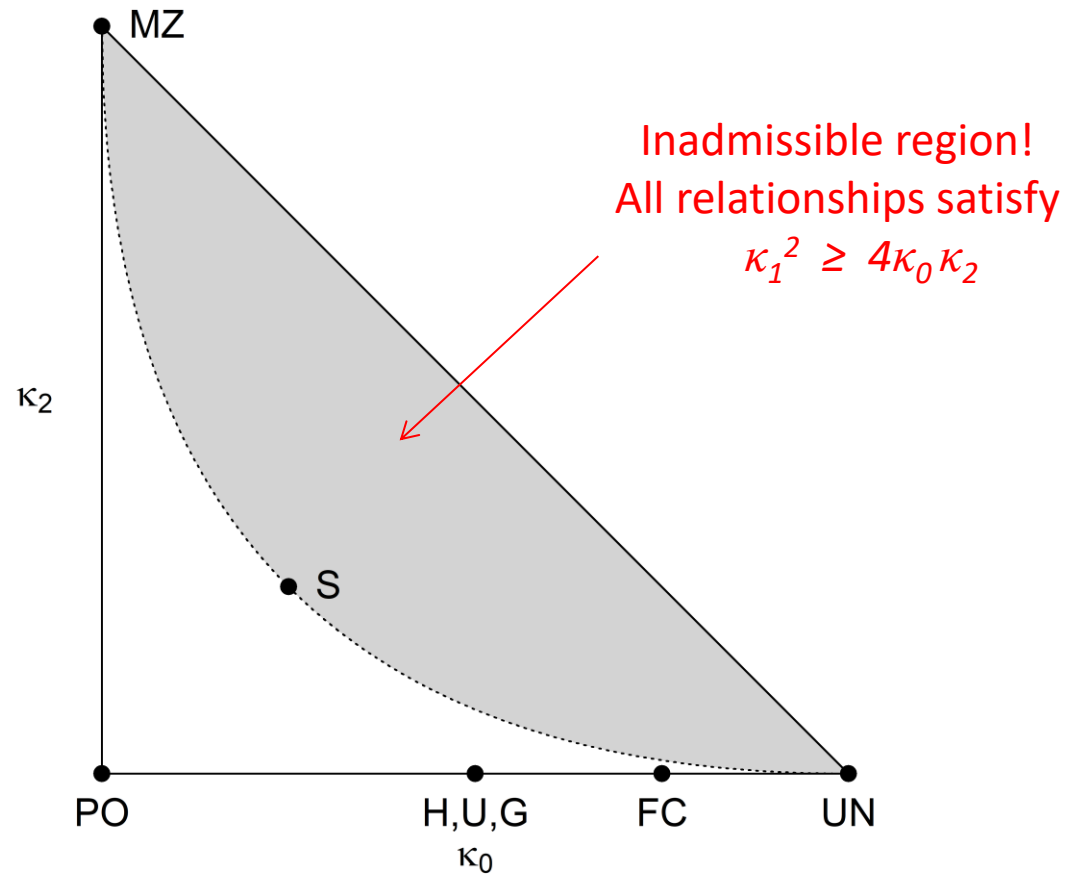
> x = nuclearPed(2)
> plot(x)

> kinship(x, ids = 3:4)
[1] 0.25

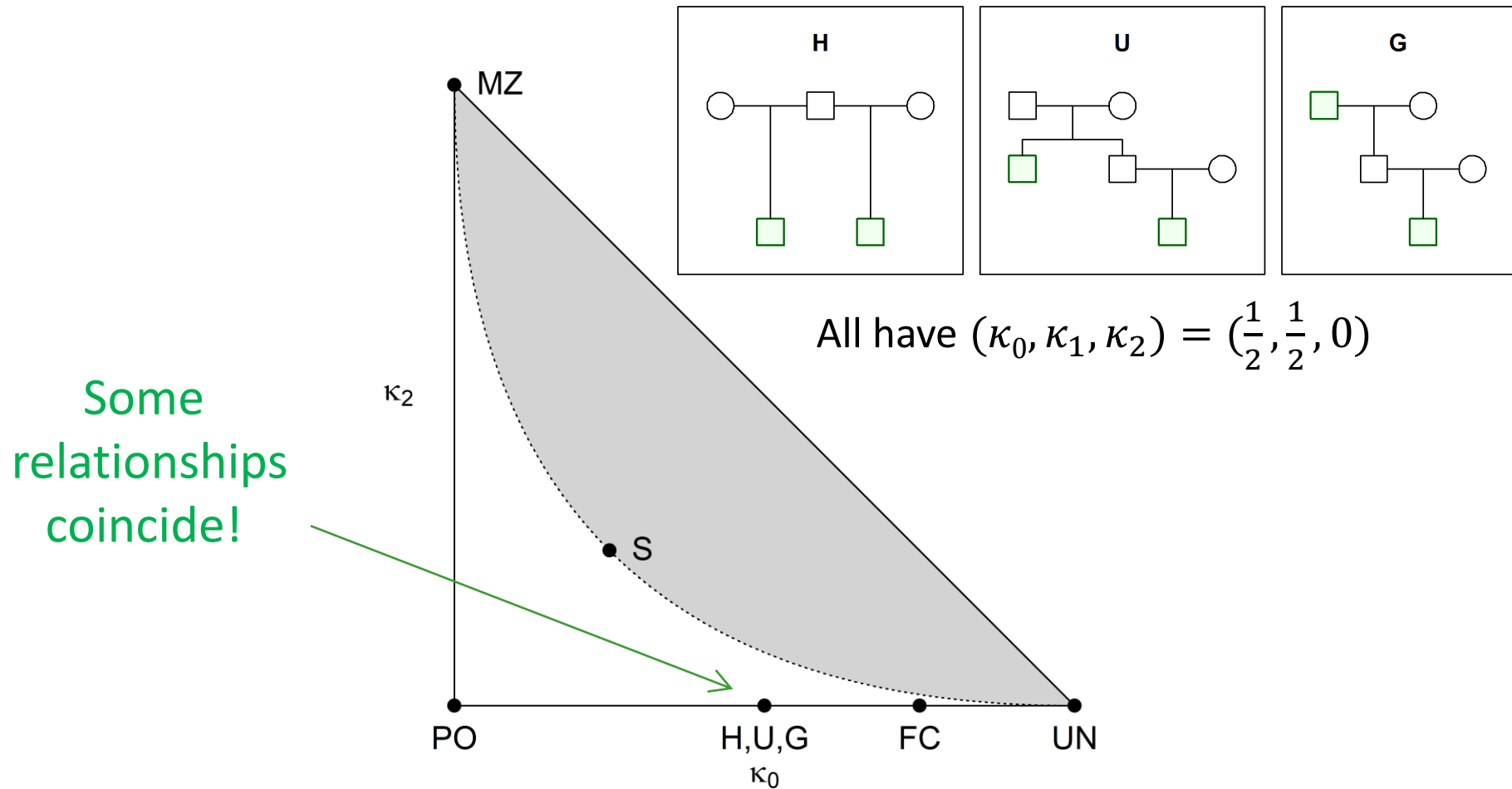
> kappaIBD(x, ids = 3:4)
[1] 0.25 0.50 0.25

> k = kappaIBD(x, ids = 3:4)
> showInTriangle(k)
```

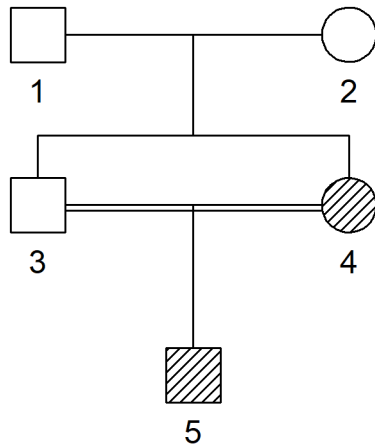
# Quirks of the relatedness triangle: 1



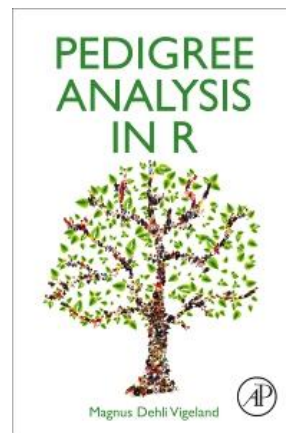
## Quirks of the relatedness triangle: 2



# A word of caution

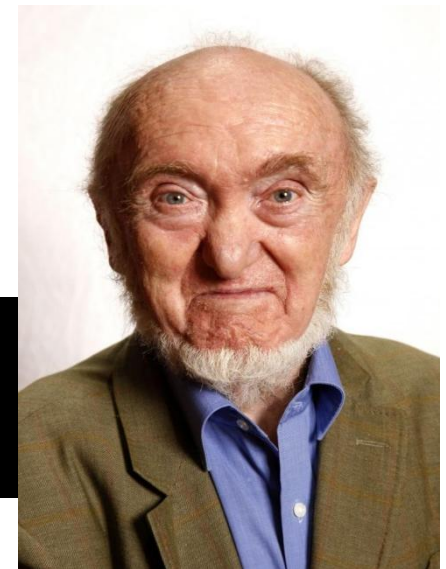


$\kappa$  is only defined for non-inbred individuals.  
For the whole story, we need 9 coefficients!



Jacquard's identity coefficients

Not suitable for a basic course ...

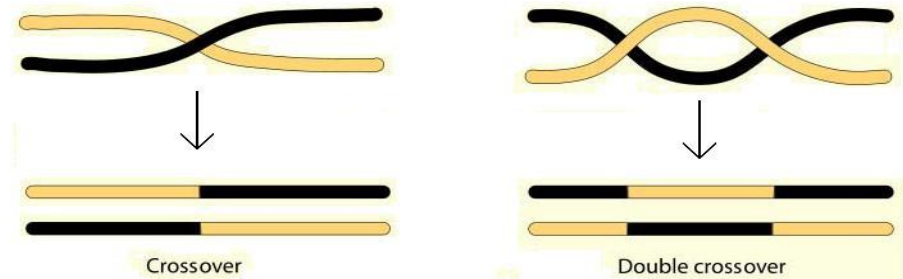
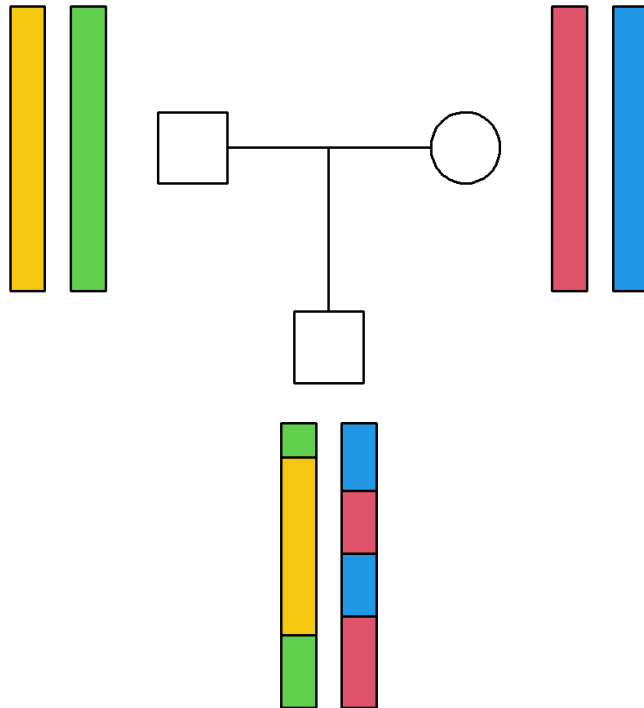


Albert Jacquard  
(1925 - 2013)

## Part II: Measures of relatedness

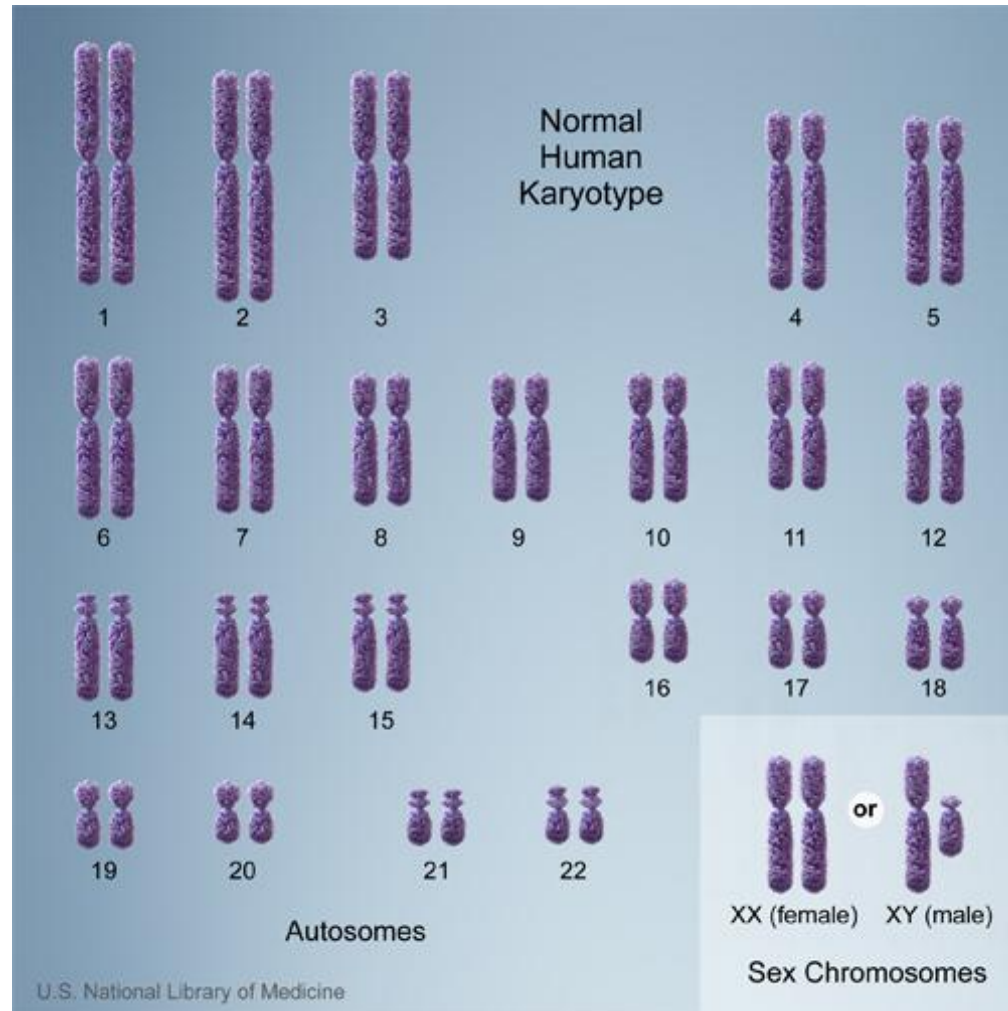
### **Realised relatedness**

# Meiotic recombination

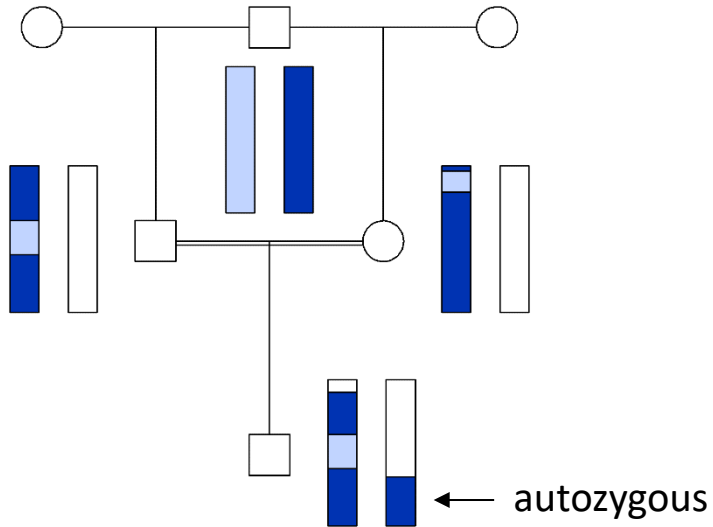


- **Genetic distance** between two loci:  
= average # crossovers/ meiosis
- Units:
  - 1 Morgan (M) = 1 crossover per meiosis
  - 1 centiMorgan (cM) = 0.01 M
- The human genome: Ca 30 Morgan

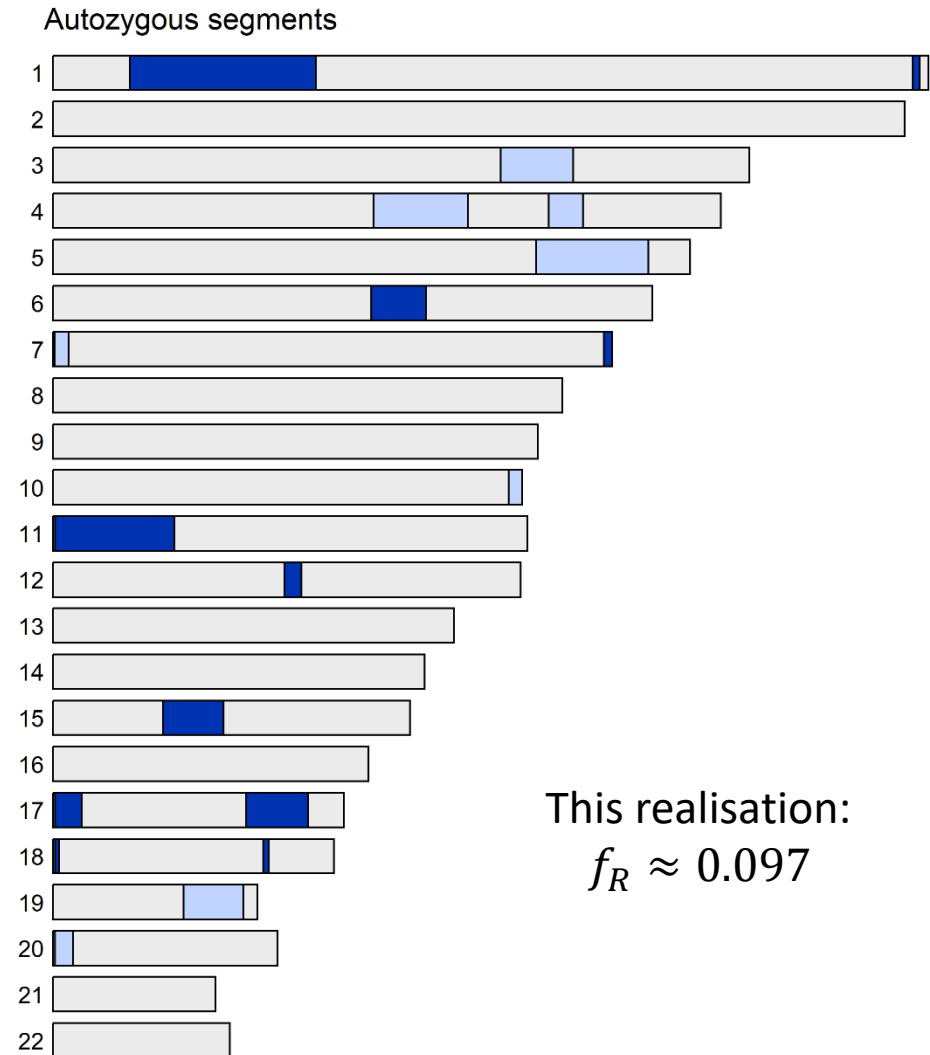
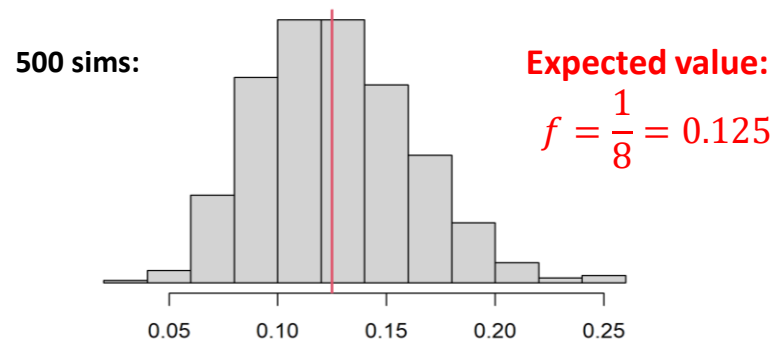
# Rule of thumb: One crossover per chromosome arm



# Realised inbreeding



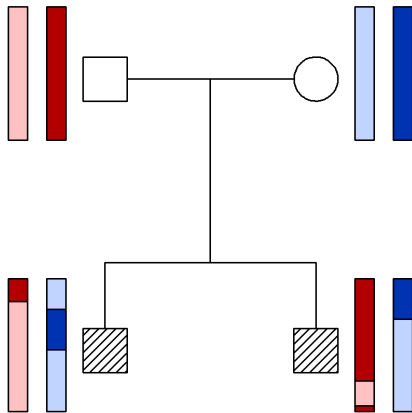
$f_R$  = autozygous fraction of the genome



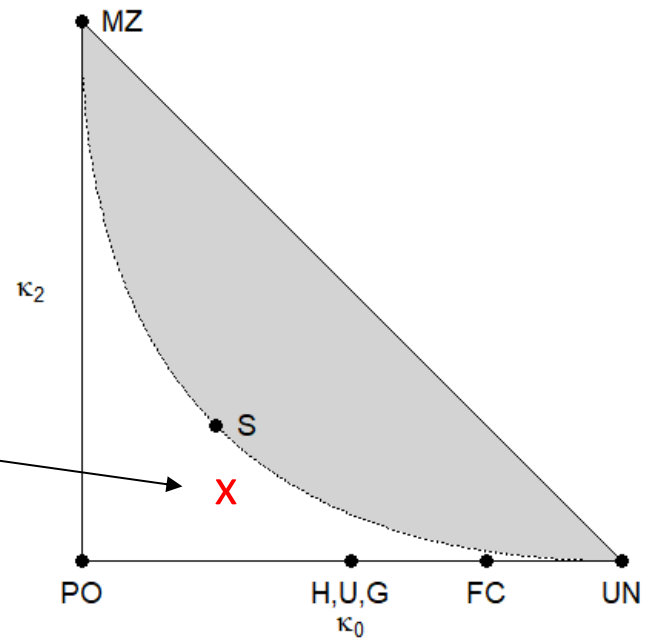


# Realised IBD coefficients

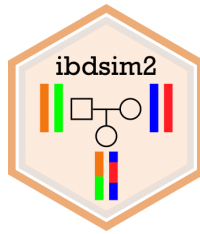
1000 simulations



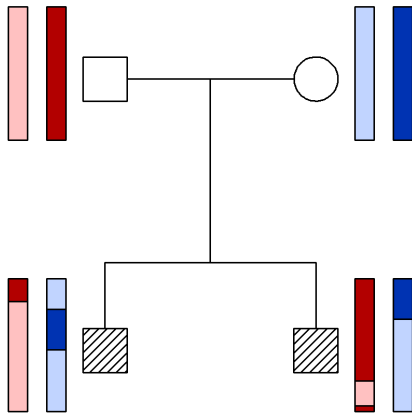
*Realised* IBD coefficients:  
Proportions of genome with IBD = 0, 1, 2



# Variation in realised IBD coefficients



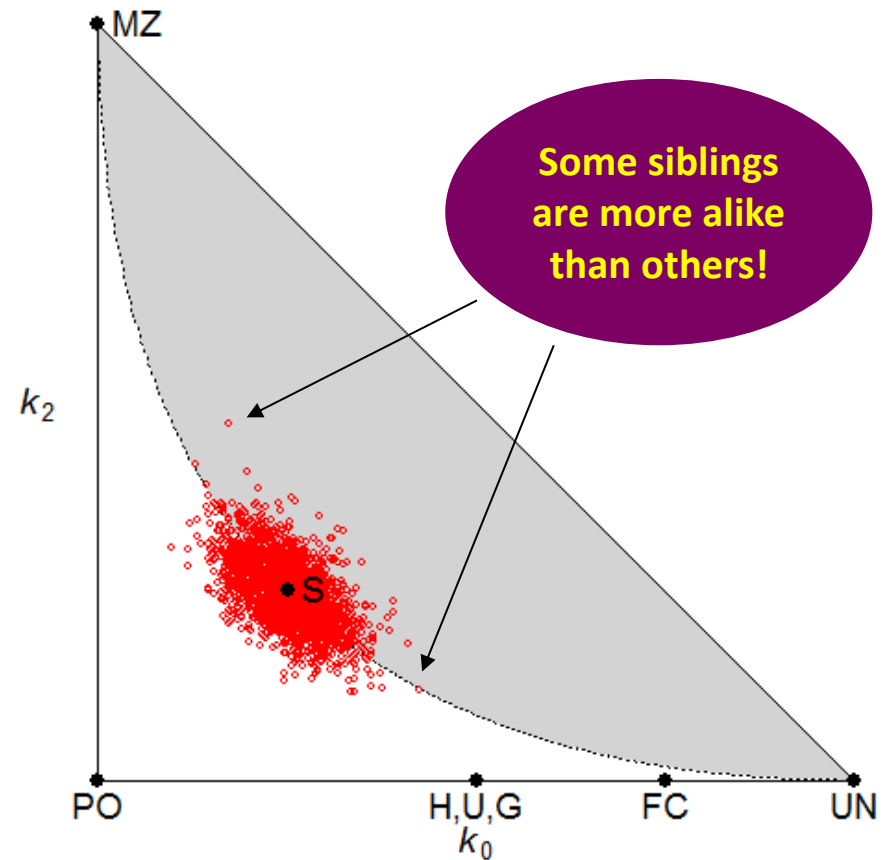
1000 simulations



```
> library(ibdsim2)

> x = nuclearPed(2)
> s = ibdsim(x, N = 1000)
> k = realisedKappa(s, ids = 3:4)

> ribd::showInTriangle(k)
```



# Variation depends on the genome



Human:

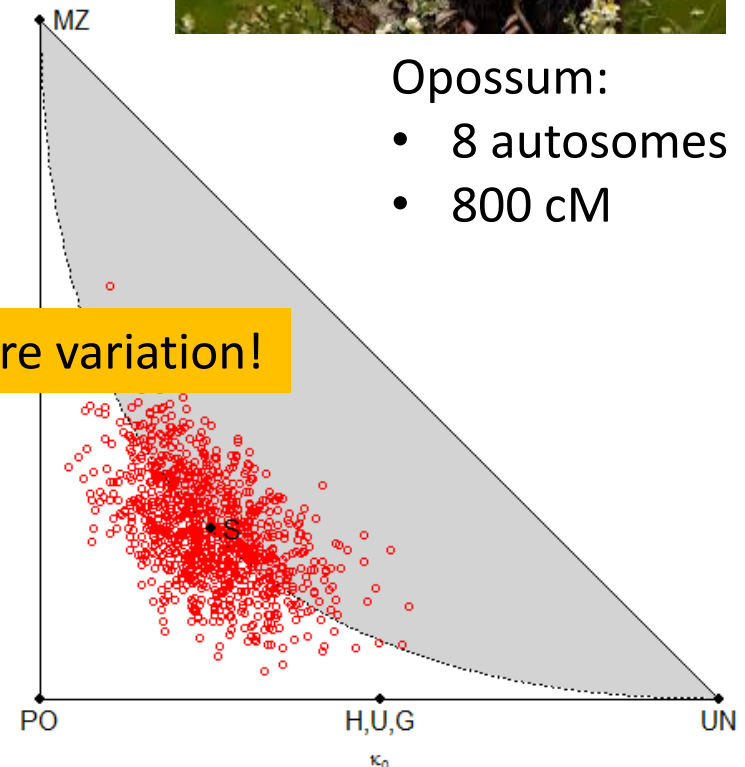
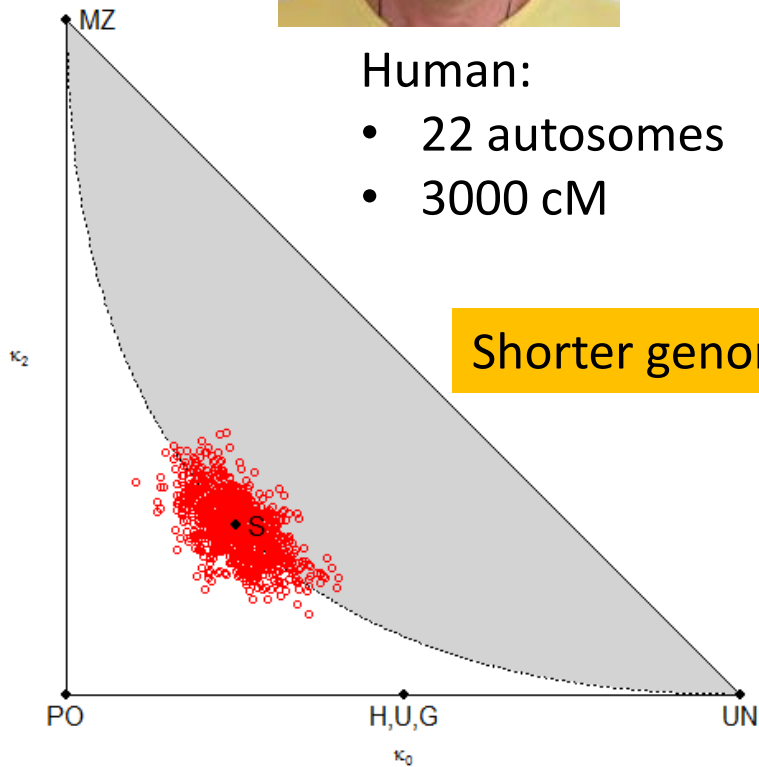
- 22 autosomes
- 3000 cM



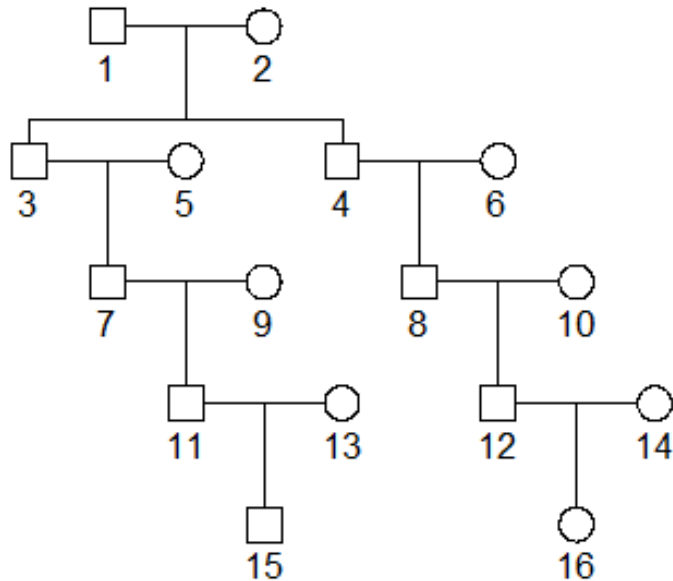
Opossum:

- 8 autosomes
- 800 cM

Shorter genome = more variation!



# The probability of zero IBD



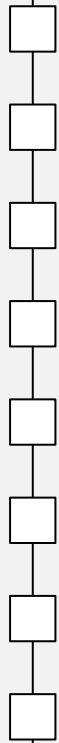
## Third cousins

Expected fraction with IBD = 1:

$$k_1 = \frac{1}{64}$$

N'th cousins	$P(\text{zero IBD})$
first	0.0 %
second	0.0 %
third	1.5 %
fourth	28 %
fifth	67 %

A pedigree relation does **not** imply a genetic relation!



Mainly for  
self study!

Part III: Crash course in



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



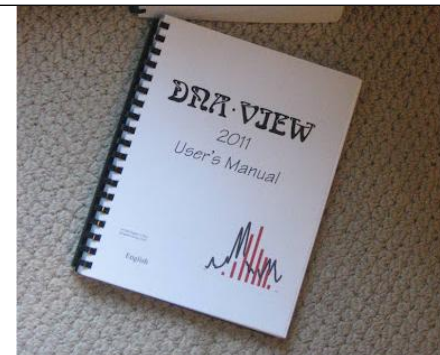
- A framework for statistical computing
  - calculator
  - data handling and numerical analysis
  - flexible plotting
  - programming language
  - external packages
    - anyone can make one
    - thousands!

## 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!**

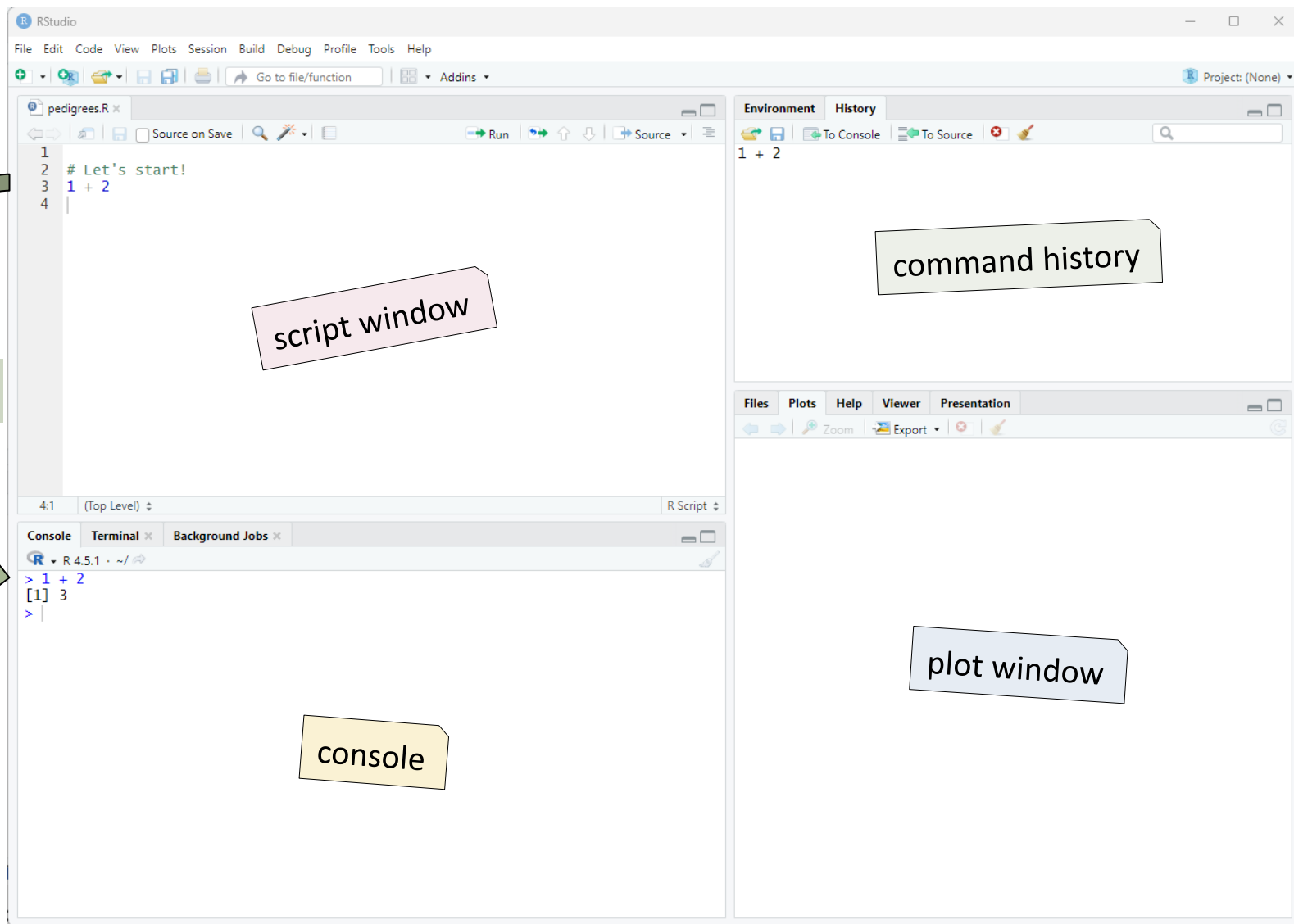
It's not that hard.

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



To get started,  
open RStudio!







# 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 logical 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
```

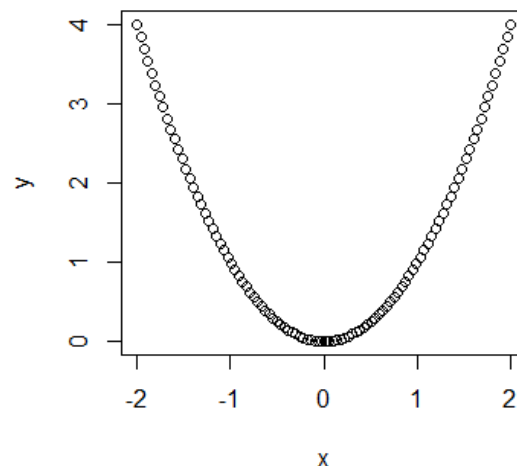
```
$ok
```

```
[1] -1
```

# Basic plots

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

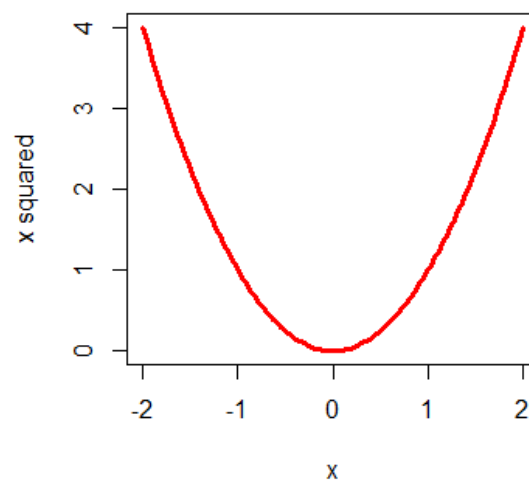
```
> x = seq(-2, 2, length = 100)
> y = x^2
> plot(x, y)
```



Many options to play with...

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

My plot



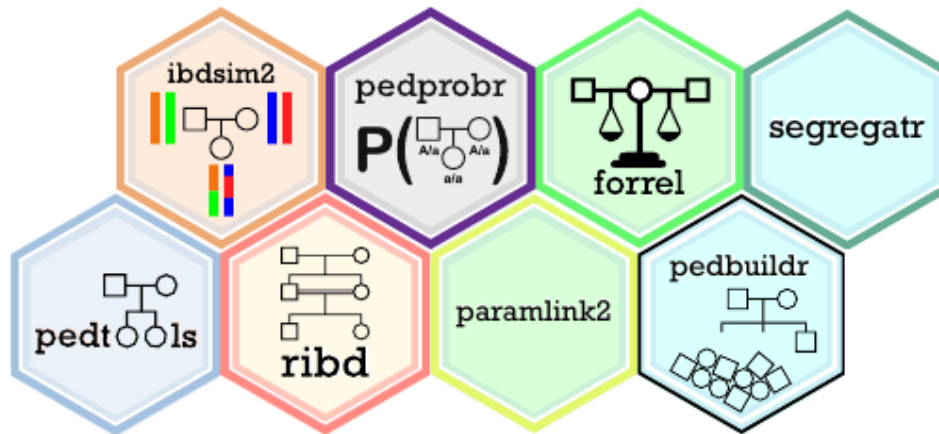
# R stuff skipped in this brief introduction

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



# The *pedsuite*:

## A collection of packages for pedigree analysis in R

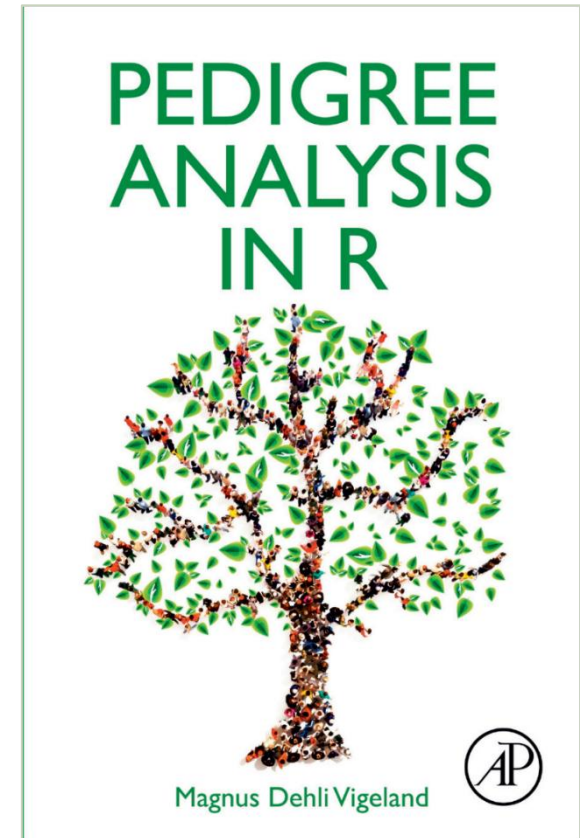


Home page:

<https://magnusdv.github.io/pedsuite>

Source code + documentation:

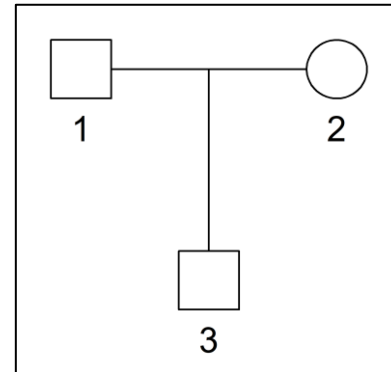
<https://github.com/magnusdv>





# Your first pedigree

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



# Some useful functions

## Create: basic

- singleton
- nuclearPed
- linearPed
- halfSibPed
- cousinPed
- halfCousinPed

## Create: complex

- ancestralPed
- doubleCousins
- quadHalfFirstCousins
- fullSibMating
- randomPed

## Manipulate

- addSon
- addDaughter
- 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:

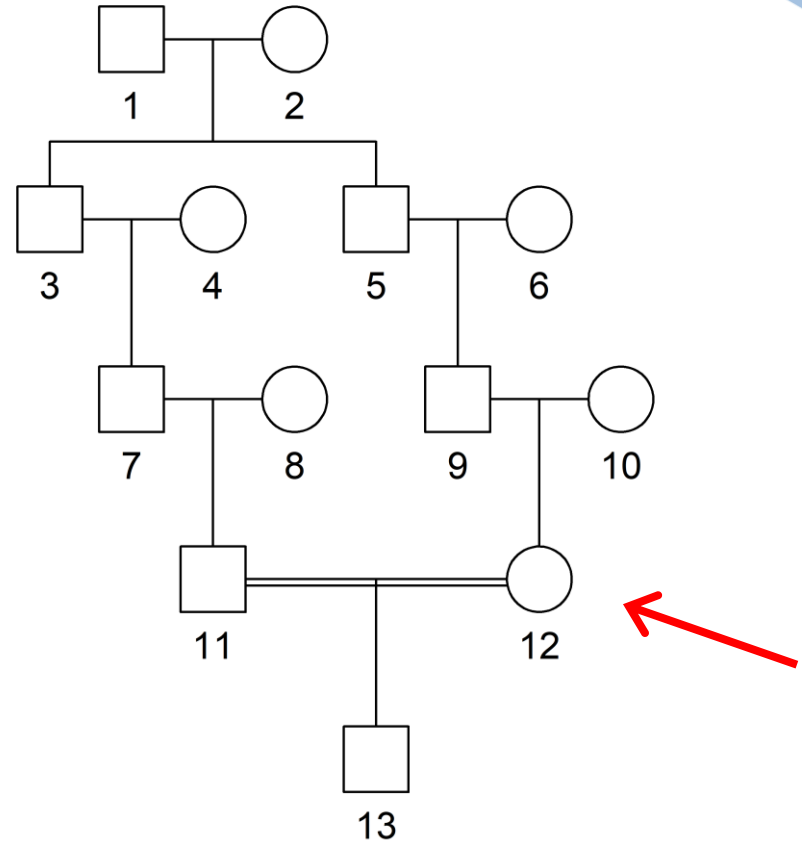
```
> x = swapSex(x, 12)
> plot(x)
```

Add inbred child

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

## Remember

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

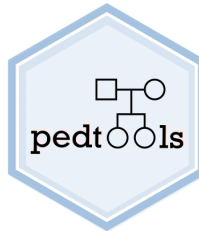


## Shortcut command for this pedigree

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

# The pipe |>

Introduced in R 4.1



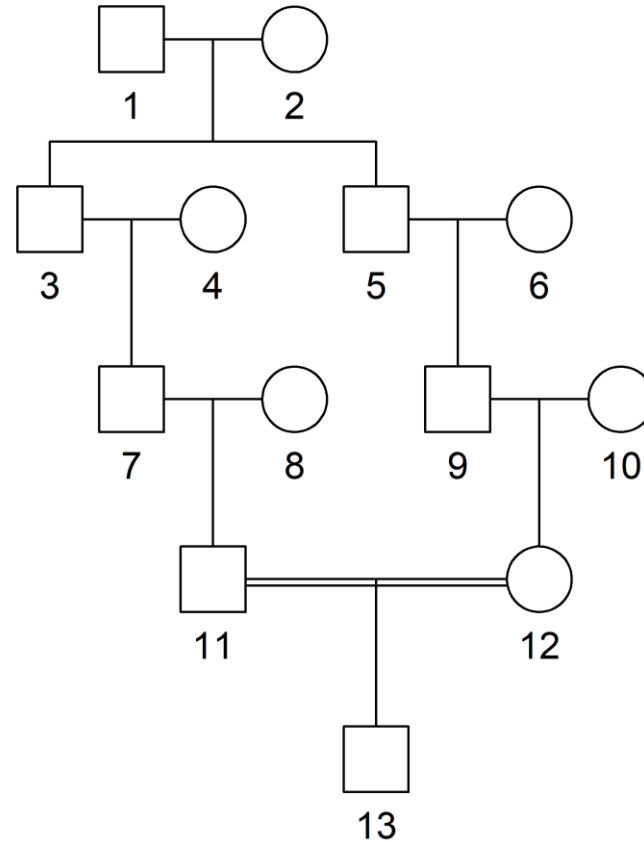
Instead of ...

```
> x = cousinPed(2)
> x = swapSex(x, 12)
> x = addSon(x, parents = 11:12)
```

... we can write

```
> x = cousinPed(2) |>
  swapSex(12) |>
  addSon(parents = 11:12)
```

Feeds the previous result  
into the next function!



Your turn: Exercises!

