A pedigree-transmission likelihood for multiplex families

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Chapter 1

Introduction and Background

Chapter 2

Bayesian Networks

Chapter 3

Pedigrees

(casting a pedigree as a Bayesian networks)

Chapter 4

Transmission Likelihood ?

Chapter 5

Examples

Chapter 6

Conclusions

Chapter 7

Appendix

(with full code listing)

```
library("gRain")
```

```
## Warning: package 'gRain' was built under R version 4.0.5

## Loading required package: gRbase

## Warning: package 'gRbase' was built under R version 4.0.5
```

```
library("Rgraphviz")
```

```
## Loading required package: graph

## Loading required package: BiocGenerics

## Warning: package 'BiocGenerics' was built under R version 4.0.5

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: grid
```

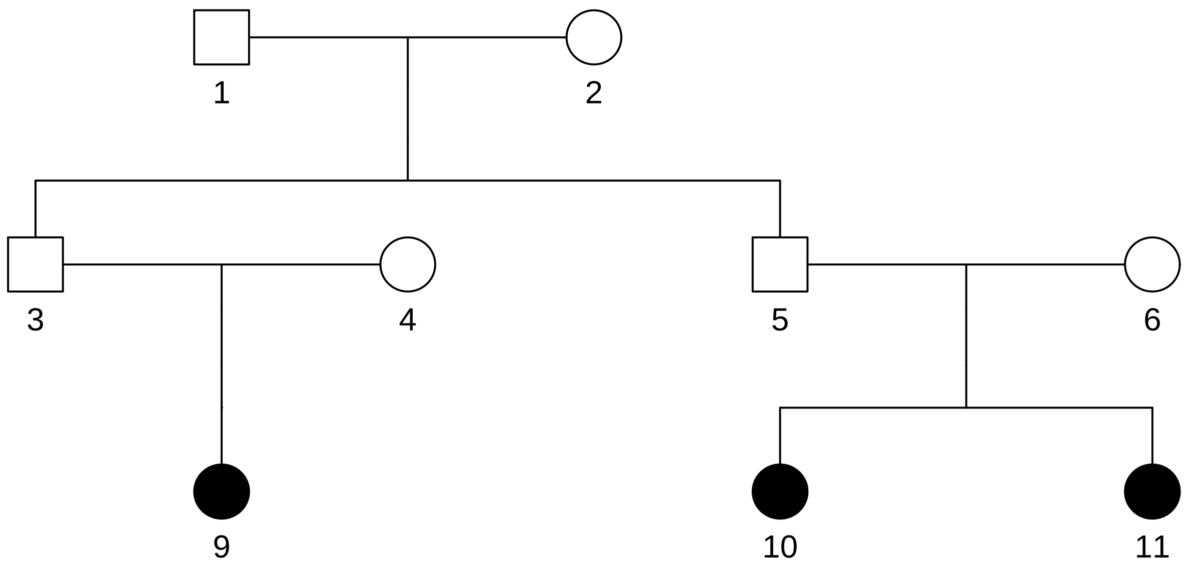
```
library(kinship2)
```

```
## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.0.5

## Loading required package: quadprog
```

Part1: gRain package exercise with toy pedigree

The first section is to use functions from the gRain package to calculate the probability that the three affected individuals ID9, ID10, ID11 received a rare variant(RV) from the founder(ID1) given that (1) the founder carries the rare variant and (2) the rare variant is transmitted with probability 3/4 for each transmission from the founder down the lines of descent to the affected individuals.

```
# Plot the toy pedigree using pedigree function from kinship2 package

id = c(1, 2, 3, 4, 5, 6, 9, 10, 11)
# Label each family member with id number
father = c(0, 0, 1, 0, 1, 0, 3, 5, 5)
# Use id number to indicate the father of each family member
mother = c(0, 0, 2, 0, 2, 0, 4, 6, 6)
# Use id number to indicate the mother of each family member
sex = c(1, 2, 1, 2, 1, 2, 2, 2, 2)
# 1 represents male, 2 represents female
affected = c(0, 0, 0, 0, 0, 0, 1, 1, 1)
# affected specifies which family member is affeccted by the particular disease
# 0 represents individual doesn't have the disease,
# 1 represents individual has the disease

toyPed = as.data.frame(cbind(id, father, mother, sex, affected))

toyPed_ob = pedigree(toyPed$id, toyPed$father, toyPed$mother, toyPed$sex, affected = toyPed$affected)
# Use built in function pedigree from kinship2 package to construct
# the toy pedigree indicated above
plot(toyPed_ob)
```



```r
p = 3/4
# In the toy pedigree case, we assumed the transmission probability is 3/4
# from the founder down the lines of descent to the affected individuals

copy = c("0", "1", "2")
# number of variants levels: 0, 1, 2

# conditional prob of child in the first entry of the triplet
#given parents in the second and third entries of triplet(27 combinations)

child_1 = c(1, 0, 0, 1/4, 3/4, 0, 0, 3/4, 0)
# 000 100 200 010 110 210 020 120 220
child_2 = c(1/4, 3/4, 0, 1/16, 3/8, 9/16, 0, 1/4, 3/4)
# 001 101 201 011 111 211 021 121 221
child_3 = c(0, 1, 0, 0, 1/4, 3/4, 0, 0, 1)
# 002 102 202 012 112 212 022 122 222
child = c(child_1, child_2, child_3)


# Specify conditional probability tables with values as given in child vector

# Conditional probability tables for founders
ID_1 = cptable(~id1, values = rep(1/3, 3), levels = copy)
ID_2 = cptable(~id2, values = rep(1/3, 3), levels = copy)
ID_4 = cptable(~id4, values = rep(1/3, 3), levels = copy)
ID_6 = cptable(~id6, values = rep(1/3, 3), levels = copy)
# We don't know the values for the founder since they don't have parents in our
# pedigree(they come from other pedigrees), so unknown transmission prob, we
# can't simply use the child vector above.

# Conditional probability tables for non-founders
ID_3.12 = cptable(~id3 + id1 + id2, values = child, levels = copy)
ID_5.12 = cptable(~id5 + id1 + id2, values = child, levels = copy)
ID_9.34 = cptable(~id9 + id3 + id4, values = child, levels = copy)
ID_10.56 = cptable(~id10 + id5 + id6, values = child, levels = copy)
```

```
ID_11.56 = cptable(~id11 + id5 + id6, values = child, levels = copy)
# Note the + operator does not commute (i.e. the order if the variables matter)
# + operator is merely as a seperator of the variables followed by the order
# ~child node + parents nodes

# Create an intermediate representations of CPTs:
plist_pedigree <-
    compileCPT(list(ID_1, ID_2, ID_3.12, ID_4, ID_5.12,
                    ID_6, ID_9.34, ID_10.56, ID_11.56))
# For example, we have CPTs for id3
plist_pedigree$id3
```

```
## , , id2 = 0
##
##    id1
## id3 0    1 2
##   0 1 0.25 0
##   1 0 0.75 1
##   2 0 0.00 0
##
## , , id2 = 1
##
##    id1
## id3    0      1    2
##   0 0.25 0.0625 0.00
##   1 0.75 0.3750 0.25
##   2 0.00 0.5625 0.75
##
## , , id2 = 2
##
##    id1
## id3 0    1 2
##   0 0 0.00 0
##   1 1 0.25 0
##   2 0 0.75 1
```

```
# Create the network from the list of CPTs
gin1 = grain(plist_pedigree)
summary(gin1)
```

```
## Independence network: Compiled: TRUE Propagated: FALSE
##  Nodes : chr [1:9] "id1" "id2" "id3" "id4" "id5" "id6" "id9" "id10" "id11"
##  Number of cliques:                 5
##  Maximal clique size:               3
##  Maximal state space in cliques:   27
```

```
# We assumed rare variant only comes in id1
bn1 = setEvidence(gin1, nodes = c("id1", "id2", "id4", "id6"),
                  states = c("1", "0", "0", "0") )
f1 = setFinding(bn1, nodes = c("id9", "id10", "id11"),
                states = c("1", "1", "1"))
pFinding(f1)
```

```
## [1] 0.002929688
```

```
# the network can be queried:
querygrain(bn1, nodes = c("id9", "id10", "id11"), type = "marginal")
```

```
## $id9
## id9
##      0      1      2
## 0.4375 0.5625 0.0000
##
## $id10
## id10
##      0      1      2
## 0.4375 0.5625 0.0000
##
## $id11
## id11
##      0      1      2
## 0.4375 0.5625 0.0000
```

```
# The joint distribution of the three affected individuals
# ID9, ID10, ID11 received a rare  variant(RV)
# from the founder(ID1) can be obtained:
querygrain(bn1, nodes = c("id9", "id10", "id11"), type = "joint")
```

```
## , , id11 = 0
##
##    id10
## id9         0          1 2
##   0 0.1298828 0.06152344 0
##   1 0.1669922 0.07910156 0
##   2 0.0000000 0.00000000 0
##
## , , id11 = 1
##
##    id10
## id9          0         1 2
##   0 0.06152344 0.1845703 0
##   1 0.07910156 0.2373047 0
##   2 0.00000000 0.0000000 0
##
## , , id11 = 2
##
##    id10
## id9 0 1 2
##   0 0 0 0
##   1 0 0 0
##   2 0 0 0
```

Part 2: Bayesian network construction

Function BNcreate() takes two arguments

- Kinship2 pedigree object

- Transmission probability of rare variant Kinship2 Pedigree object with their plot (should I move this part down?)

```
# sample pedigree data from Kinship2 package
data("sample.ped")

pedAll = pedigree(id = sample.ped$id, dadid = sample.ped$father,
                  momid = sample.ped$mother, sex = sample.ped$sex,     affected=cbind(sample.ped$affecte
print(pedAll)
```

```
## Pedigree list with 55 total subjects in 2 families
```

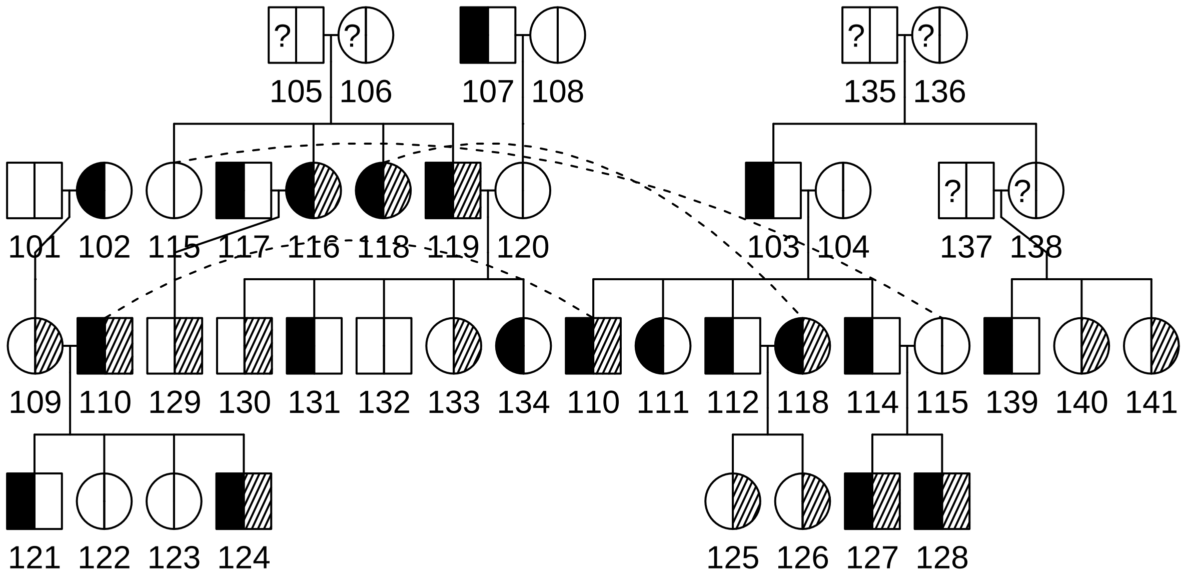
```
ped1basic = pedAll["1"]
ped2basic = pedAll["2"]
print(ped1basic)
```

```
## Pedigree object with 41 subjects, family id= 1
## Bit size= 46
```

```
print(ped2basic)
```

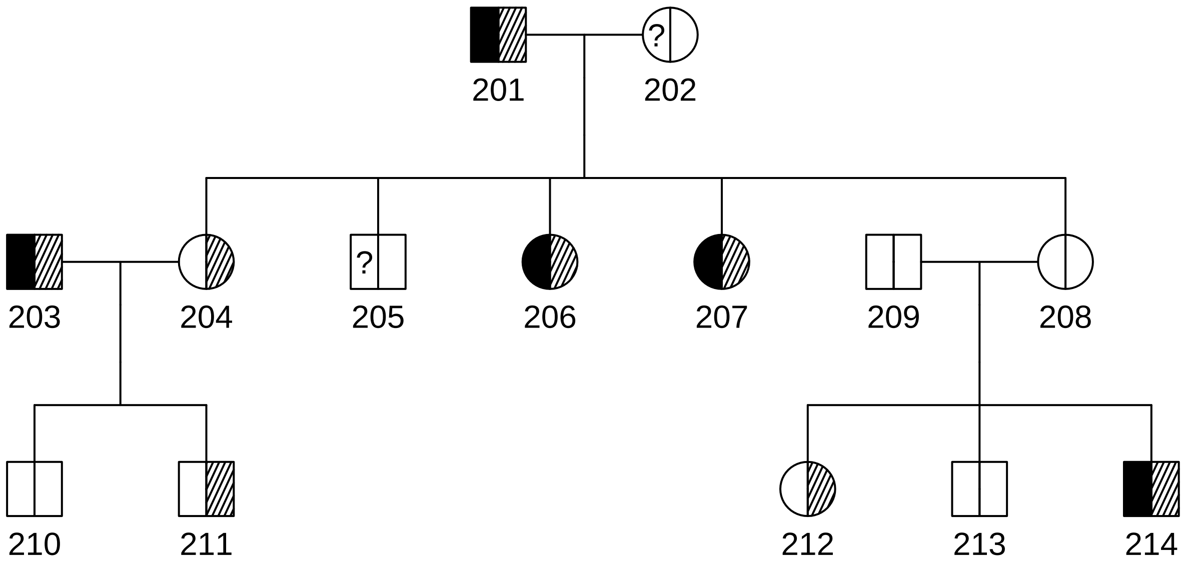
```
## Pedigree object with 14 subjects, family id= 2
## Bit size= 16
```

```
####
pedfile_total = sample.ped # two pedigrees
pedfile = pedfile_total[42:55,]
pedfile_2 = pedfile_total[1:41,]
####
plot(ped1basic)
```



```
## Did not plot the following people: 113
```

```
plot(ped2basic)
```



```
# Individual with affected labelled NA was marked as?

pedfile
```

```
##    ped  id father mother sex affected avail
## 42   2 201      0      0   1        1     1
## 43   2 202      0      0   2       NA     0
## 44   2 203      0      0   1        1     1
## 45   2 204    201    202   2        0     1
## 46   2 205    201    202   1       NA     0
## 47   2 206    201    202   2        1     1
## 48   2 207    201    202   2         1      1
## 49   2 208    201    202   2         0      0
## 50   2 209      0      0   1         0      0
## 51   2 210    203    204   1         0      0
## 52   2 211    203    204   1         0      1
## 53   2 212    209    208   2         0      1
## 54   2 213    209    208   1         0      0
## 55   2 214    209    208   1         1      1
```

Write a function BNcreate() to create a Bayesian network

```
BNcreate = function(pedfile, tau){

  ## conditional prob of child in the first entry of the triplet
  ## given parents in the second and third entries of triplet with transmission
  ## probability tau
  geno_prob = c(
    # 000 100 200 010 110 210 020 120 220
    1, 0, 0, 1-tau, tau, 0, 0, 1, 0,
    # 001 101 201 011 111 211 021 121 221
    1-tau, tau, 0, (1-tau)^2, 2*tau*(1-tau), tau^2, 0, 1-tau, tau,
    # 002 102 202 012 112 212 022 122 222
    0, 1, 0, 0, 1-tau, tau, 0, 0, 1
  )

   # Construct the CPTs for founders
   # Founders: columns of father or mother are labelled as 0.

   founders = which(pedfile[, "father"] == 0) # the row numbers of founders
   founders_vec = rep(0, length(founders))
   # store the id number of founders (for user configuration)
   founders_cpt = list()
   # store the CPTs of founders in a list, which is denoted as [[i]]
   for (i in 1:length(founders)) {
     id = pedfile[founders[i], "id"]
     founders_vec[i] = id
     node = cptable(c(id), values = rep(1/3, 3), levels = copy)
     # CPTs of founders
     founders_cpt[i] = list(node)
   }

   # Construct the CPTs for non-founders

   pedfile_c = pedfile[-c(founders), ]
   # Pedigree data after removing founders
   family_ids = list()
   # Store the c(child, father, mother) for debugging
   nonfounders_cpt = list()
   # Store the CPTs for non-founders(for debugging)
   for (i in 1:nrow(pedfile_c)) {
     family_ids[i] = list(c(pedfile_c[i, 'id'], pedfile_c[i, "father"],
                            pedfile_c[i, "mother"]))
     c = pedfile_c[i, 'id'] # id numbers of child
     f = pedfile_c[i, "father"] # id numbers of father
```

```
      m = pedfile_c[i, "mother"] # id numbers of mother
      node_nf = cptable(c(c, f, m), values = geno_prob, levels = copy)
      # CPTs for non-founders
      nonfounders_cpt[i] = list(node_nf)
    }

    plist = compileCPT(founders_cpt, nonfounders_cpt)
    gin = grain(plist)
    # Create the Bayesian network from the CPTs of both founders and non-founders

    return(gin)
    #return(list(plist, founders_vec, family_ids))

}


gin = BNcreate(pedfile, 3/4)
# Create and return the Bayesian network from the list of CPTs (family Ped file)
# with transmission probability 3/4
```

Part3: Likelihood Function construction

Function likehd takes 3 arguments:

- pedfile: Kinship2 pedigree object specifying ped structure

- tau: Transmission probability of rare variant

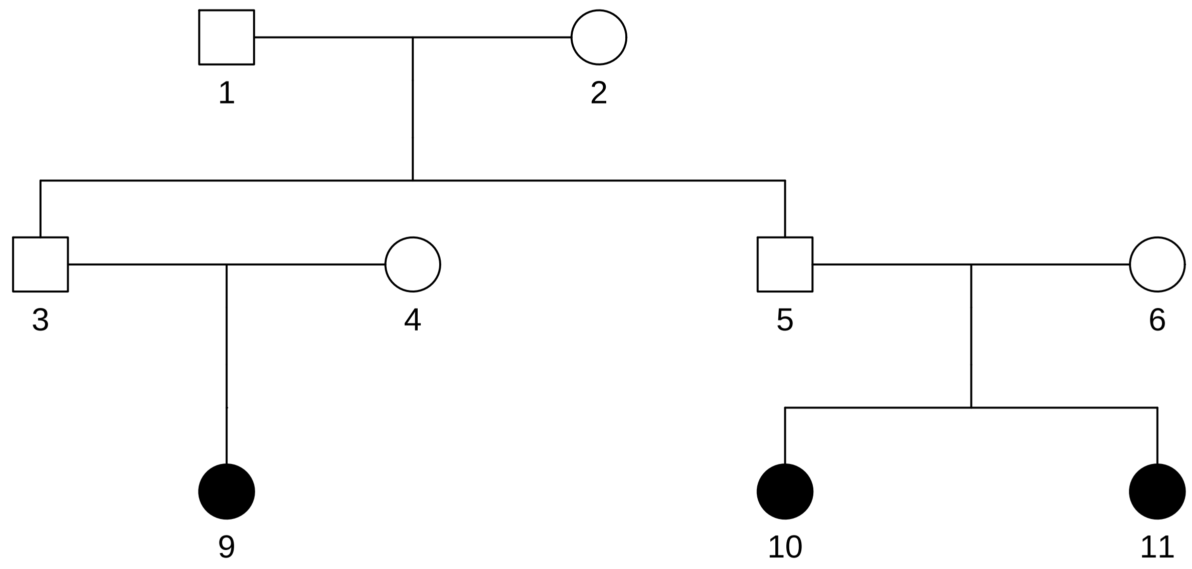
- config: Rare variant configuration of affected individuals Note: User specified config vector only contains two numbers, 0 and 1. 0 means individual has the disease but do not carry the rare variant. 1 means individual has the disease and carries the rare variant. The positions of numbers correspond to the affected individuals by order in the family pedigree. For example, if the affected individuals in a pedigree are id(1, 3, 6, 7, 14) and user specified configuration vectors is config c("1", "0", "1", "1", "1"), which means id 1, 6, 7, 14 both are affected and carry the rare variants.

```
likehd = function(pedfile, tau, config){

  gin = BNcreate(pedfile, tau)
  # Bayesian networks of the specified pedigree with transmission prob tau
  likelihood = 0

  founders = which(pedfile[, "father"] == 0)
  # find the row numbers of founders
  founders_vec = rep(0, length(founders))
  # store the id number of founders (for user configuration)
  for (i in 1:length(founders)) {
    id = pedfile[founders[i], "id"] # id number of founders
    founders_vec[i] = as.character(id)
  }
  affected = which(pedfile[, "affected"] == 1)
  # find the row numbers of affected individuals
  affected_vec = rep(0, length(affected))
  # affected_vec is used to store the id numbers of affected individuals
  for (i in 1:length(affected)) {
    id = pedfile[affected[i], "id"]
    # id number of affected individuals
    affected_vec[i] = as.character(id)
  }

  print(affected_vec)
  # We assume rare variant comes into a pedigree on exactly one founder at a time
  # loop through all the probabilities that each founder carries the RV
  for (i in 1:length(founders)) {
    state = rep("0", length(founders))
    # vector state specifies which founder introduces the rare variant
    state[i] = "1"
    # let founder i introduces the RV into pedigree
    #print(state) # 1 means carrying the rare variant
    bn = setEvidence(gin, nodes = founders_vec, states = state)
    # update the Bayseian networks and set founder i introduces the rare variant
    bn_find = setFinding(bn, nodes = affected_vec, states = config)
    # calculate P(configuration | founder i introduced the rv)
    # configuration is specified by user as config
    p = pFinding(bn_find)
    # The probability of observing the finding is obtained with pFinding()
    #print(p)
    likelihood = p*1/length(founders)+likelihood
  }

  return(likelihood)
}
```

Part4: Test likelihood function with different pedigrees

4.1 Toy Pedigree

```
plot(toyPed_ob)
```



```
toy_config = c("1", "1", "1")

v = likehd(toyPed, 3/4, toy_config)
```

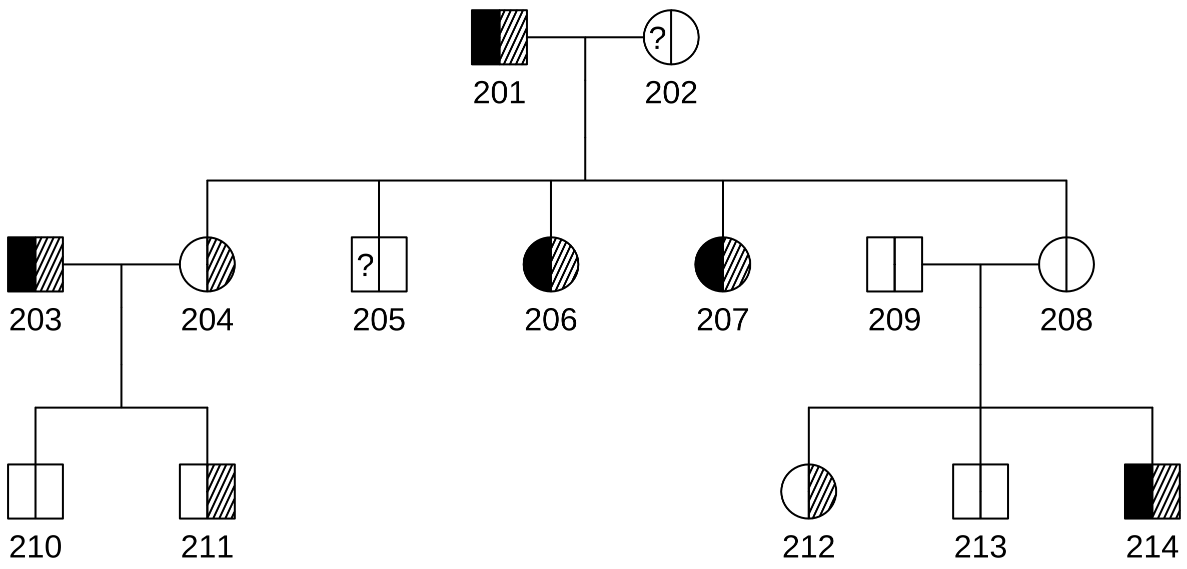
```
## [1] "9"  "10" "11"
```

```
tail(v, 1)
```

```
## [1] 0.001464844
```

4.2 More complex pedigree with differnt configurations

```
plot(ped2basic)
```



```r
config1 = c("1", "0", "1", "1", "1")
likehd(pedfile, 3/4, config1)
```

```
## [1] "201" "203" "206" "207" "214"
```

```
## [1] 0.001953125
```

```r
config2 = c("0", "1", "1", "0", "1")
likehd(pedfile, 3/4, config2)
```

```
## [1] "201" "203" "206" "207" "214"
```

```
## [1] 0.0006510417
```

```r
config3 = c("1", "1", "0", "0", "0")
likehd(pedfile, 3/4, config3)
```

```
## [1] "201" "203" "206" "207" "214"
```

```
## [1] 0.004026813
```

Part5: Plot likelihood function VS tau values

Use likelihood function with different value of tau, plot likelihood function VS tau values. Check the curve with different configurations

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.