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*.

```
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
```

Part1: gRain package exercise with toy pedigree

```
## RV is transmitted with prob 3/4 for each transmission from the founder down the lines of descent to
p = 3/4


## use 3 possible levels (0,1,2) represent genotype
copy = c("0", "1", "2")
# conditional prob of child in the first entry of the triplet given parents in the second and third ent



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

length(child)
```

```
## [1] 27
```

```
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)

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)




#Specification of a network
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))
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
```

Part 2:Bayesian network construction

Function BNcreate() takes two arguments

-Kinship2 pedigree object

-transmission probability of rare variant Kinship2 Pedigree object with their plot

```
library(kinship2)
```

```
## Loading required package: Matrix

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

## Loading required package: quadprog
```

```
data("sample.ped")

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

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

```
ped1basic = pedAll["1"]
ped2basic = pedAll["2"]

####
pedfile_total = sample.ped
pedfile_2 = pedfile_total[1:41,]
pedfile = pedfile_total[42:55,]
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
```

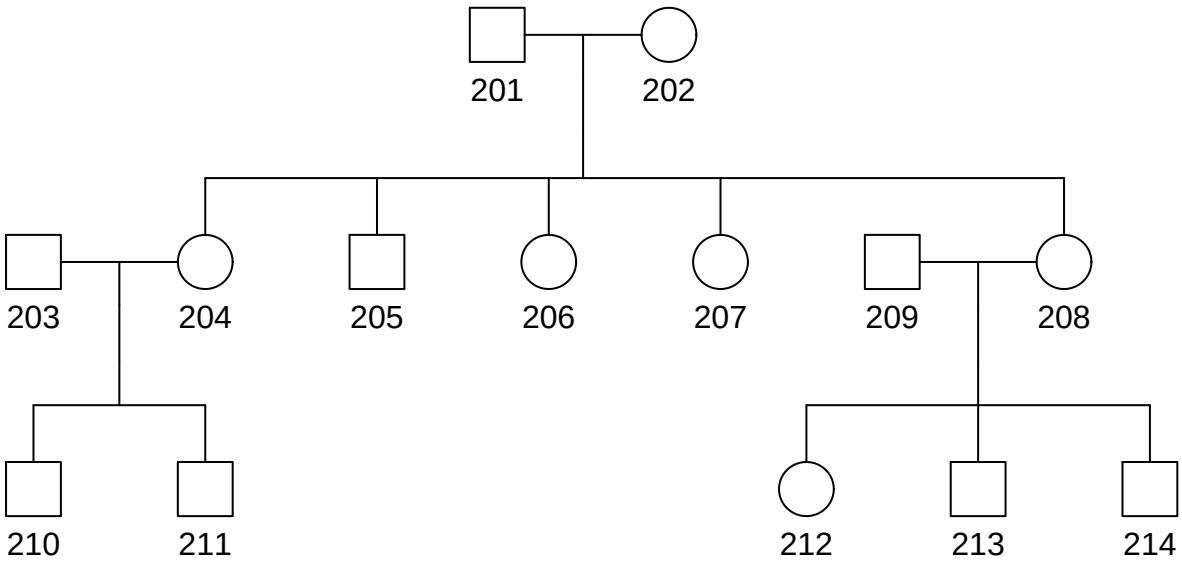
```
####
print(ped1basic)
```

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

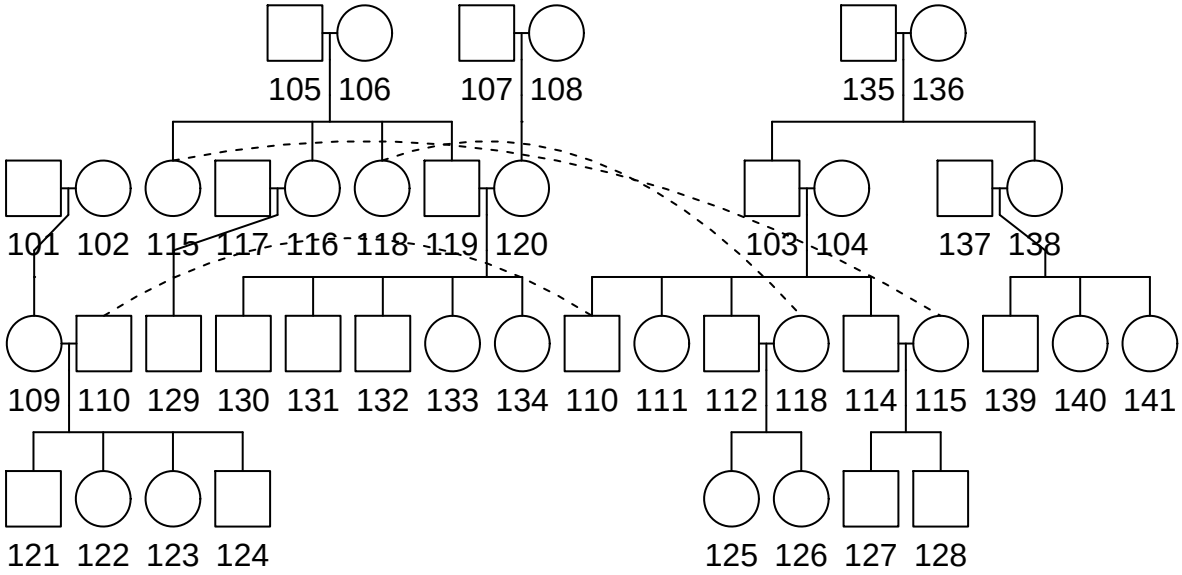
```
print(ped2basic)
```

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

```
plot(ped2basic)
```




```
plot(ped1basic)
```




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

Write a function BNcreate() to create a Bayesian network

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

```
  # if columns of father and mother are lablled 0, which means they are founders, create cptables for f
   founders = which(pedfile[, "father"] == 0) # the row number of founders
   founders_vec = rep(0, length(founders)) # store the id numbder of founders (for debugging)
   founders_cpt = list() # store the cptable of founders, 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)
     # this id node will be applied to setEvidence function later
     founders_cpt[i] = list(node)
   }

   pedfile_c = pedfile[-c(founders), ] # dataset removing founders
   family_ids = list() ## for debugging
   nonfounders_cpt = list()
   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']
     f = pedfile_c[i, "father"]
     m = pedfile_c[i, "mother"]
     node_nf = cptable(c(c, f, m), values = geno_prob, levels = copy)
     #debug for the c,f,m here, they are specified more than once
     nonfounders_cpt[i] = list(node_nf)
   }

   plist = compileCPT(founders_cpt, nonfounders_cpt)
   gin = grain(plist)
   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)
```

Part3: Likelihood Function construction

Function likehd takes 3 arguments:

- Kinship2 pedigree object specifying ped structure

- Transmission probability of rare variant

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

  gin = BNcreate(pedfile, tau)
  likelihood = 0
```

```
  founders = which(pedfile[, "father"] == 0) # the row number of founders
  founders_vec = rep(0, length(founders)) # store the id numbder of founders
  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)
  affected_vec = rep(0, length(affected))
  # store the id number of affected individuals
  for (i in 1:length(affected)) {
    id = pedfile[affected[i], "id"]
    affected_vec[i] = as.character(id)
  }

  print(affected_vec)
  for (i in 1:length(founders)) {
    state = rep("0", length(founders))
    state[i] = "1" # Assume only one founder introduces the rv at a time
    print(state) # 1 means which founder introduces the rare variant
    bn = setEvidence(gin, nodes = founders_vec, states = state)
    # which founder introduces the rv, assuming only one introduced at a time
    bn_find = setFinding(bn, nodes = affected_vec, states = config)
    # calculate P(configuration | which founder introduced the rv)
    p = pFinding(bn_find)
    # The probability of observing the finding is obtained with pFinding()
    print(p)
    likelihood = p*1/length(founders)+likelihood
  }

  return(likelihood)
}
```

- Rare variant configuration of affected individuals

Part4: Test likelihood function with different pedigrees

4.1 Toy Pedigree

```
id = c(1, 2, 3, 4, 5, 6, 9, 10, 11)
father = c(0, 0, 1, 0, 1, 0, 3, 5, 5)
mother = c(0, 0, 2, 0, 2, 0, 4, 6, 6)
sex = c(1, 2, 1, 2, 1, 2, 2, 2, 2)
affected = c(0, 0, 0, 0, 0, 0, 1, 1, 1)
toyPed = cbind(id, father, mother, sex, affected)
toy_config = c("1", "1", "1")

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

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

```
## [1] "1" "0" "0" "0"
## [1] 0.002929688
## [1] "0" "1" "0" "0"
## [1] 0.002929688
## [1] "0" "0" "1" "0"
## [1] 0
## [1] "0" "0" "0" "1"
## [1] 0

## [1] 0.001464844
```

4.2 More complex pedigree with differnt configurations

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

```
## [1] "201" "203" "206" "207" "214"
## [1] "1" "0" "0" "0"
## [1] 0.00390625
## [1] "0" "1" "0" "0"
## [1] 0.00390625
## [1] "0" "0" "1" "0"
## [1] 0
## [1] "0" "0" "0" "1"
## [1] 0

## [1] 0.001953125
```

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

```
## [1] "201" "203" "206" "207" "214"
## [1] "1" "0" "0" "0"
## [1] 0.001302083
## [1] "0" "1" "0" "0"
## [1] 0.001302083
## [1] "0" "0" "1" "0"
## [1] 0
## [1] "0" "0" "0" "1"
## [1] 0

## [1] 0.0006510417
```

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

```
## [1] "201" "203" "206" "207" "214"
## [1] "1" "0" "0" "0"
## [1] 0.0003375772
```

```
## [1] "0" "1" "0" "0"
## [1] 0.0003375772
## [1] "0" "0" "1" "0"
## [1] 0.01234568
## [1] "0" "0" "0" "1"
## [1] 0.00308642

## [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*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.