Graphical Models and Bayesian Networks – useR!2015 tutorial

Therese Graversen

Department of Mathematical Sciences, University of Copenhagen

30 June 2015

Bayesian Networks

Models for discrete variables

Imagine that we are interested in the distribution of a set of discrete random variables that each take a finite number of values.

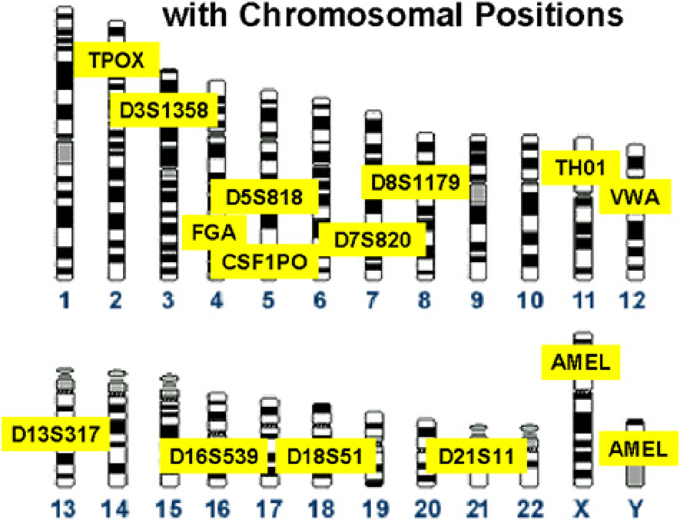
A Bayesian network is a convenient framework for

- Specifying the joint distribution (a model)
- efficiently computing marginal and conditional probabilities.

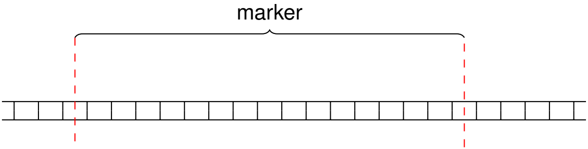
Example: Forensic identification using DNA

- Paternity cases
- Forensic identification in mass disasters
- Samples of poor quality or mixed from many people.

13 CODIS Core STR Loci with Chromosomal Positions



STR marker: An identifiable area (locus) on a chromosome



Allele: The DNA sequence at a marker
A or B
(In practice there are 10-20 possible alleles for a marker)

Genotype: Unordered pair of alleles
(A, A), (A, B), or (B, B).

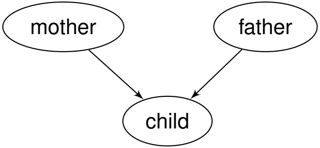
The genotype of a person at a specific marker is a random variable with state space $\{AA, AB, BB\}$.

We are interested in the joint distribution of genotypes for a group of people.

The Bayesian network representation

A Bayesian Network represents the joint distribution by

1. a *DAG*, where each node corresponds to a variable
2. a collection of *conditional probability tables (CPTs)*.



The DAG specifies a factorisation of the joint distribution

$$p(\mathtt{m}, \mathtt{f}, \mathtt{c}) = p(\mathtt{m})p(\mathtt{f})p(\mathtt{c} \mid \mathtt{m}, \mathtt{f}).$$

Importantly,

- the joint distribution can be evaluated as the product of CPTs
- each factor depends only on a subset of the variables

Inheritance

A child inherits one allele from each parent independently.

The parent's two alleles have equal probability of being passed on to the child.

Each combination has probability 1/4; some lead to the same genotype for the child.

	A	A
A	AA	AA
A	AA	AA

	A	A
A	AA	AA
B	AB	BB

	A	B
A	AA	AB
B	AB	BB

Conditionally on the parents, the distribution of the child is

```
##        mother   AA                  AB                  BB
##        father   AA   AB   BB   AA   AB   BB   AA   AB   BB
## child
## AA            1.00 0.50 0.00 0.50 0.25 0.00 0.00 0.00 0.00
## AB            0.00 0.50 1.00 0.50 0.50 0.50 1.00 0.50 0.00
## BB            0.00 0.00 0.00 0.00 0.25 0.50 0.00 0.50 1.00
```

Conditional probability tables (CPT) for a child

```
prob <- function(child, mother, father){
      child <- strsplit(child, "")[[1]]
        mother <- strsplit(mother, "")[[1]]
        father <- strsplit(father, "")[[1]]
        ## Probability of inheriting allele a from genotype gt
        P <- function(a, gt)((a == gt[1]) + (a == gt[2]))/2
        if (child[1] != child[2]){
              P(child[1], mother)*P(child[2], father) +
                    P(child[1], father)*P(child[2], mother)
            } else {
                  P(child[1], mother)*P(child[2], father)
                }
    }
gts <- c("AA", "AB", "BB")
tab <- expand.grid(child=gts, mother = gts, father = gts,
                    stringsAsFactors=FALSE)
tab$prob <- mapply(prob, tab$child, tab$mother, tab$father)
## We save the probabilities for use in CPTs
inheritance <- tab$prob
```

CPTs are arrays of probabilities

```
head(tab)
```

```
##   child mother father prob
## 1    AA     AA     AA  1.0
## 2    AB     AA     AA  0.0
## 3    BB     AA     AA  0.0
## 4    AA     AB     AA  0.5
## 5    AB     AB     AA  0.5
## 6    BB     AB     AA  0.0
```

```
c.mf <- xtabs(prob ~ ., tab)
ftable(c.mf, row.vars = "child")
```

```
##       mother   AA                AB                BB
##       father   AA   AB   BB   AA   AB   BB   AA   AB   BB
## child
## AA           1.00 0.50 0.00 0.50 0.25 0.00 0.00 0.00 0.00
## AB           0.00 0.50 1.00 0.50 0.50 0.50 1.00 0.50 0.00
## BB           0.00 0.00 0.00 0.00 0.25 0.50 0.00 0.50 1.00
```

Marginal distribution of a genotype

If alleles A and B occur in the population with frequencies $(0.3, 0.7)$, then the distribution of genotypes AA, AB, and BB is

```
gtprobs <- dbinom(0:2, size = 2, prob = c(0.3, 0.7))
```

Assuming that the person's 2 alleles are sampled independently from the population.

Building the network

We can specify the Bayesian network via a list of CPTs.

Each CPT can be specified via e.g.

- `array`
- `parray`
- `cptable`

Building the network

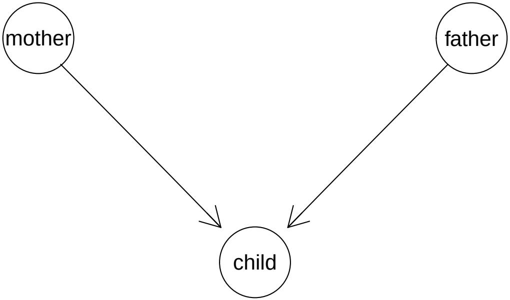
```
mother <- cptable(~mother, values = gtprobs, levels=gts)
(father <- cptable(~father, values = gtprobs, levels=gts))

## {v,pa(v)} : chr "father"
##    <NA>
## AA 0.49
## AB 0.42
## BB 0.09

(child <- cptable(~child | mother + father,
                  values = inheritance, levels = gts))

## {v,pa(v)} : chr [1:3] "child" "mother" "father"
##    <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## AA    1  0.5    0  0.5 0.25  0.0    0  0.0    0
## AB    0  0.5    1  0.5 0.50  0.5    1  0.5    0
## BB    0  0.0    0  0.0 0.25  0.5    0  0.5    1
```

Building the network

```
## A list of CPTs
(cptlist <- compileCPT(list(child, mother, father)))

## CPTspec with probabilities:
##  P( child | mother father )
##  P( mother )
##  P( father )

## Bayesian Network
(trio <- grain(cptlist))

## Independence network: Compiled: FALSE Propagated: FALSE
##   Nodes: chr [1:3] "child" "mother" "father"
```

Building the network

```
plot(trio)
```



Joint (marginal) distribution of a set of variables

```
## Marginal distribution of the father's genotype
querygrain(trio, nodes = "father")

## $father
## father
##   AA   AB   BB
## 0.49 0.42 0.09

## Joint distribution of mother and child
ftable(querygrain(trio, nodes = c("child", "mother"),
                  type = "joint"),
       col.vars = "child")

##        child    AA    AB    BB
## mother
## AA            0.343 0.147 0.000
## AB            0.147 0.210 0.063
## BB            0.000 0.063 0.027
```

Joint (conditional) distribution of a set of variables

```
## Conditional distribution of the father given mother and chi
ftable(querygrain(trio, nodes=c("father", "child", "mother"),
                  type = "conditional"),
       col.vars = "father")

##              father   AA   AB   BB
## child mother
## AA    AA            0.70 0.30 0.00
##       AB            0.70 0.30 0.00
##       BB             NaN  NaN  NaN
## AB    AA            0.00 0.70 0.30
##       AB            0.49 0.42 0.09
##       BB            0.70 0.30 0.00
## BB    AA             NaN  NaN  NaN
##       AB            0.00 0.70 0.30
##       BB            0.00 0.70 0.30
```

Evidence

If we observe a configuration of some of the variables, this can be entered as *evidence*.

Then the network gives the

- conditional distribution given the evidence
- marginal probability of the evidence

Joint (conditional) distribution of a set of variables

```
## Network with evidence entered
trio_ev <- setEvidence(trio, nodes=c("child", "mother"),
                       states = c("AB", "BB"))
## p(father | child = AB, mother = BB)
querygrain(trio_ev, nodes="father")

## $father
## father
##  AA  AB  BB
## 0.7 0.3 0.0

## Removing all entered evidence
trio_ev <- retractEvidence(trio_ev)
## p(father)
querygrain(trio_ev, nodes="father")

## $father
## father
##   AA   AB   BB
## 0.49 0.42 0.09
```

Probability of a configuration of a set of variables

Method 1: Get the entire joint distribution and find your configuration:

```
querygrain(trio, nodes=c("child", "mother"),
              type = "joint")
```

```
##       mother
## child    AA    AB    BB
##    AA 0.343 0.147 0.000
##    AB 0.147 0.210 0.063
##    BB 0.000 0.063 0.027
```

Method 2: Enter the configuration as evidence and get the normalising constant.

```
tr <- setEvidence(trio, nodes=c("child", "mother"),
                  evidence = c(child="AB", mother="BB"))
```

```
pEvidence(tr)
```

```
## [1] 0.063
```

Simulation

We can simulate directly from the distribution that the Bayesian network represents:

```
## Prior distribution
simulate(trio, 3)

##   child mother father
## 1    AA     AB     AA
## 2    BB     BB     AB
## 3    AA     AA     AA

## Posterior after observing child and mother
simulate(trio_ev, 3)

##   child mother father
## 1    BB     BB     BB
## 2    AB     AB     AB
## 3    AA     AA     AA
```

Example: Paternity testing

A mother with genotype BB has a child with genotype AB. She claims that Mr X, who has genotype AB, is the father of her child.

The *evidence* in this case could be the observed genotypes of the mother and the child.

We compare the probability of the evidence under two alternative hypotheses:

H_1: Mr X is the father
vs.
H_2: Some unknown man is the father

We need to compute

$$LR = \frac{\mathrm{P}(\mathtt{c} = AB, \mathtt{m} = BB \mid H_1)}{\mathrm{P}(\mathtt{c} = AB, \mathtt{m} = BB \mid H_2)} = \frac{\mathrm{P}(\mathtt{c} = AB, \mathtt{m} = BB \mid \mathtt{f} = AB)}{\mathrm{P}(\mathtt{c} = AB, \mathtt{m} = BB)}$$

Example: Paternity testing

$$LR = \mathrm{P}(\mathtt{c} = AB, \mathtt{m} = BB \mid \mathtt{f} = AB) / \mathrm{P}(\mathtt{c} = AB, \mathtt{m} = BB)$$

```
## P(m = BB, c = AB, f = AB)
p.fmc <- pEvidence(setEvidence(trio, evidence = list(
                                        mother = "BB",
                                        child = "AB",
                                        father = "AB")))
## P(f = AB)
p.f <- pEvidence(setEvidence(trio, evidence = list(
                                        father = "AB")))
L.H1 <- p.fmc/p.f
## P(m = BB, c = AB)
L.H2 <- pEvidence(setEvidence(trio, evidence = list(
                                        mother = "BB",
                                        child = "AB")))
## Likelihood ratio comparing Mr X vs unknown person.
L.H1/L.H2

## [1] 0.7142857
```

The likelihood ratio is smaller than 1, so the evidence does not point to Mr X being the father.

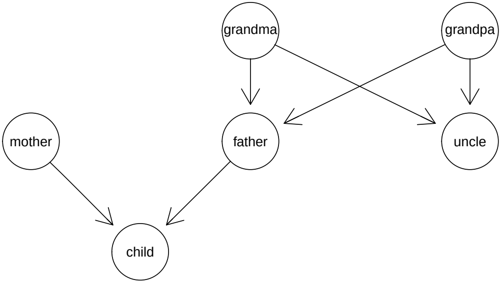
Conditional independence

In a Bayesian Network, any variable is conditionally independent of its non-descendants given its parents; e.g.

$$\texttt{uncle} \perp\!\!\!\perp (\texttt{mother}, \texttt{father}, \texttt{child}) \mid (\texttt{grandma}, \texttt{grandpa})$$

$$\texttt{mother} \perp\!\!\!\perp (\texttt{grandma}, \texttt{grandpa}, \texttt{father}, \texttt{uncle})$$

We use that to construct the network.



$$p(\texttt{c}, \texttt{m}, \texttt{f}, \texttt{un}, \texttt{gm}, \texttt{gf}) = p(\texttt{c} \mid \texttt{m}, \texttt{f}) p(\texttt{m}) p(\texttt{f} \mid \texttt{gm}, \texttt{gf}) p(\texttt{un} \mid \texttt{gm}, \texttt{gf}) p(\texttt{gf}) p(\texttt{gm})$$

Missing father, but the uncle is available

```
## p(child | mother, father)
c.mf <- parray( c("child", "mother", "father"),
                  levels = rep(list(gts), 3),
                  values = inheritance)
## p(father | grandma, grandpa)
f.gmgf <- parray( c("father", "grandma", "grandpa"),
                    levels = rep(list(gts), 3),
                    values = inheritance)
## p(uncle | grandma, grandpa)
u.gmgf <- parray(c("uncle", "grandma", "grandpa"),
                    levels = rep(list(gts), 3),
                    values = inheritance)
## p(mother)
m <- parray("mother", values = gtprobs, levels=list(gts))
## p(grandpa)
gf <- parray("grandpa", values = gtprobs, levels = list(gts))
## p(grandma)
gm <- parray("grandma", values = gtprobs, levels = list(gts))

cpt.list <- compileCPT(list(c.mf, m, f.gmgf, u.gmgf, gm, gf))
extended.family <- grain(cpt.list)
```

Practical exercises

1. Build the network `extended.family` on your own computer.
2. A mother claims that Mr X is the father of her child. Unfortunately it is not possible to get a DNA sample from Mr X, but his brother ("uncle") is willing to give a sample.

mother	AB
child	AB
uncle	AA

   What is the probability of observing this evidence, i.e. this combination of genotypes?
3. What is the conditional distribution of the father's genotype given the evidence?
4. Ignoring the genotypes of the mother and the uncle, what is the conditional distribution of the father's genotype given that the child is AB?

Behind the scenes: Local computations on a junction tree

Joint distribution of all variables

The 6 persons each has 3 possible genotypes, so the state space has $3^6 = 729$ states. This quickly grows!

```
joint <- tableListProd(cpt.list) ; ftable(joint, row.vars = 1:6)
```

```
## grandpa grandma uncle child mother father
## AA      AA      AA    AA    AA     AA        0.11764900
##                                    AB        0.00000000
##                                    BB        0.00000000
##                             AB     AA        0.05042100
##                                    AB        0.00000000
##                                    BB        0.00000000
##                             BB     AA        0.00000000
##                                    AB        0.00000000
##                                    BB        0.00000000
##                       AB    AA     AA        0.00000000
##                                    AB        0.00000000
##                                    BB        0.00000000
##                             AB     AA        0.05042100
##                                    AB        0.00000000
##                                    BB        0.00000000
##                             BB     AA        0.02160900
##                                    AB        0.00000000
##                                    BB        0.00000000
##                       BB    AA     AA        0.00000000
##                                    AB        0.00000000
##                                    BB        0.00000000
##                             AB     AA        0.00000000
##                                    AB        0.00000000
##                                    BB        0.00000000
```

The Junction tree representation

The Bayesian network and CPTs are used for specification of the model.

Computations are done using more convenient computational structure; the *junction tree*

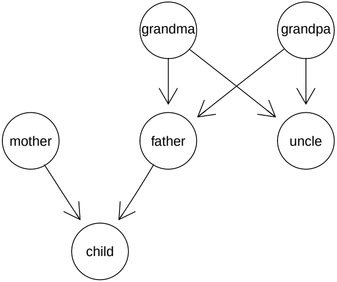
The distribution is now represented by

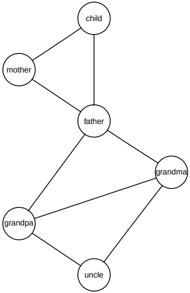
junction tree: a tree with subsets (cliques) of variables as nodes

potentials: functions of the clique variables.

Junction tree for paternity case

```
plot(extended.family)
extfam <- compile(extended.family)
plot(extfam)
plot(jTree(extfam$ug))
```

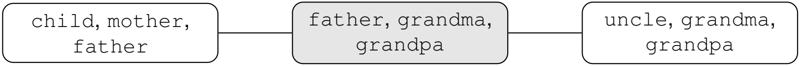




The joint distribution is the product of the potentials,

$$p(\texttt{f}, \texttt{m}, \texttt{c}, \texttt{un}, \texttt{gm}, \texttt{gf}) = q_1(\texttt{f}, \texttt{m}, \texttt{c})q_2(\texttt{f}, \texttt{gm}, \texttt{gf})q_3(\texttt{un}, \texttt{gm}, \texttt{gf})$$

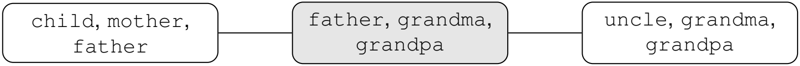
Initializing potentials



Initially we collect the CPTs according to the cliques

$$
\begin{aligned}
& p(\mathtt{f}, \mathtt{m}, \mathtt{c}, \mathtt{un}, \mathtt{gm}, \mathtt{gf}) \\
= & \underbrace{\Big(p(\mathtt{c} \,|\, \mathtt{m}, \mathtt{f}) p(\mathtt{m})\Big)}_{q_1(\mathtt{c},\mathtt{f},\mathtt{m})} \underbrace{\Big(p(\mathtt{f} \,|\, \mathtt{gm}, \mathtt{gf}) p(\mathtt{gm})\Big)}_{q_2(\mathtt{f},\mathtt{gm},\mathtt{gf})} \underbrace{\Big(p(\mathtt{un} \,|\, \mathtt{gm}, \mathtt{gf}) p(\mathtt{gf})\Big)}_{q_3(\mathtt{un},\mathtt{gm},\mathtt{gf})}
\end{aligned}
$$

Initializing potentials



The shared variables between two neighbouring cliques is their *separator*.

A slightly different factorisation also assigns a potential to each separator:

$$p(\mathtt{f}, \mathtt{m}, \mathtt{c}, \mathtt{un}, \mathtt{gm}, \mathtt{gf}) = \frac{q_1(\mathtt{f}, \mathtt{m}, \mathtt{c})q_2(\mathtt{f}, \mathtt{gm}, \mathtt{gf})q_3(\mathtt{un}, \mathtt{gm}, \mathtt{gf})}{s_1(\mathtt{f})s_2(\mathtt{gm}, \mathtt{gf})}$$

Taking the clique potentials as before, and

$$s_1(\mathtt{f}) = 1 \qquad\qquad s_2(\mathtt{gm}, \mathtt{gf}) = 1$$

we clearly did not change anything.

Similarly we can always get back to the other factorisation by “absorbing” the separator potentials into clique potentials.

Initializing potentials

```
q1 <- tabMult(m, c.mf); ftable(q1)
```

```
##                father    AA    AB    BB
## child mother
## AA    AA             0.490 0.245 0.000
##       AB             0.210 0.105 0.000
##       BB             0.000 0.000 0.000
## AB    AA             0.000 0.245 0.490
##       AB             0.210 0.210 0.210
##       BB             0.090 0.045 0.000
## BB    AA             0.000 0.000 0.000
##       AB             0.000 0.105 0.210
##       BB             0.000 0.045 0.090
```

```
q2 <- tabMult(f.gmgf, gm); ftable(q2, row.vars = c("grandma", "grandpa"))
```

```
##                  father    AA    AB    BB
## grandma grandpa
## AA      AA             0.490 0.000 0.000
##         AB             0.245 0.245 0.000
##         BB             0.000 0.490 0.000
## AB      AA             0.210 0.210 0.000
##         AB             0.105 0.210 0.105
##         BB             0.000 0.210 0.210
## BB      AA             0.000 0.090 0.000
##         AB             0.000 0.045 0.045
##         BB             0.000 0.000 0.090
```

Initializing potentials

```
q3 <- tabMult(u.gmgf, gf); ftable(q3, row.vars = c("grandma", "grandpa"))
```

```
##                  uncle    AA    AB    BB
## grandma grandpa
## AA      AA             0.490 0.000 0.000
##         AB             0.210 0.210 0.000
##         BB             0.000 0.090 0.000
## AB      AA             0.245 0.245 0.000
##         AB             0.105 0.210 0.105
##         BB             0.000 0.045 0.045
## BB      AA             0.000 0.490 0.000
##         AB             0.000 0.210 0.210
##         BB             0.000 0.000 0.090
```

```
## Separator potentials are constant 1
s1 <-  parray("father", values = 1, levels = list(gts))
s2 <-  parray(c("grandma", "grandpa"), values = 1, levels = list(gts, gts))
```

Probability propagation

Probability propagation modifies (clique- and separator-) potentials iteratively by *message passing* until they equal marginal distributions.

- Start from an intitial set of potentials
- Choose any clique to be the *root*
- Pass messages along all edges – towards the root
- Pass messages along all edges – away from the root
- All potentials now equal the marginal distributions, i.e.

$$q_1(\mathtt{f}, \mathtt{m}, \mathtt{c}) = p(\mathtt{f}, \mathtt{m}, \mathtt{c})$$
$$q_2(\mathtt{f}, \mathtt{gm}, \mathtt{gf}) = p(\mathtt{f}, \mathtt{gm}, \mathtt{gf})$$
$$q_3(\mathtt{un}, \mathtt{gm}, \mathtt{gf}) = p(\mathtt{un}, \mathtt{gm}, \mathtt{gf})$$
$$s_1(\mathtt{f}) = p(\mathtt{f})$$
$$s_2(\mathtt{gm}, \mathtt{gf}) = p(\mathtt{gm}, \mathtt{gf})$$

Message passing

Passing a message modifies a single pair of potentials: The potential for the receiving clique, and the potential for the separator between the two cliques.

Passing a message from $(\mathtt{f}, \mathtt{m}, \mathtt{c})$ to ($\mathtt{f}$, $\mathtt{gm}$, $\mathtt{gf}$) entails:

$$s_{1,\mathrm{old}}(\mathtt{f}) \leftarrow s_1(\mathtt{f})$$

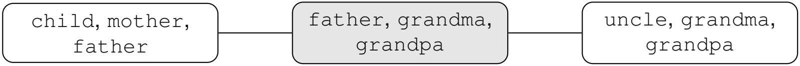
$$s_1(\mathtt{f}) \leftarrow \sum_{\mathtt{m},\mathtt{c}} q_1(\mathtt{f}, \mathtt{m}, \mathtt{c})$$

$$q_2(\mathtt{f}, \mathtt{gm}, \mathtt{gf}) \leftarrow q_2(\mathtt{f}, \mathtt{gm}, \mathtt{gf}) \frac{s_1(\mathtt{f})}{s_{1,\mathrm{old}}(\mathtt{f})}$$

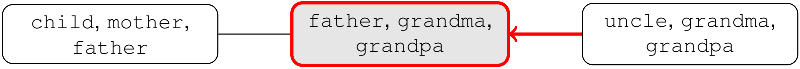
The product of potentials is unchanged:

$$\frac{q_1(\mathtt{f}, \mathtt{m}, \mathtt{c})\left(q_{2,\mathrm{old}}(\mathtt{f}, \mathtt{gm}, \mathtt{gf}) \frac{s1(\mathtt{f})}{s_{1,\mathrm{old}}(\mathtt{f})}\right) q_3(\mathtt{un}, \mathtt{gm}, \mathtt{gf})}{s1(\mathtt{f}) s_2(\mathtt{gm}, \mathtt{gf})} = \frac{q_1(\mathtt{f}, \mathtt{m}, \mathtt{c}) q_{2,\mathrm{old}}(\mathtt{f}, \mathtt{gm}, \mathtt{gf}) q_3(\mathtt{un}, \mathtt{gm}, \mathtt{gf})}{s_{1,\mathrm{old}}(\mathtt{f}) s_2(\mathtt{gm}, \mathtt{gf})}$$

Example: probability propagation



Collect evidence



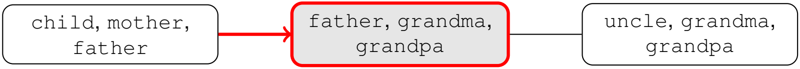

```
s1_old <- s1
(s1 <- tabMarg(q1, "father"))
```

```
## father
## AA AB BB
##  1  1  1
```

```
q2 <- tabMult(tabDiv(s1, s1_old), q2); ftable(q2)
```

```
##                 father    AA    AB    BB
## grandma grandpa                         
## AA      AA             0.490 0.000 0.000
##         AB             0.245 0.245 0.000
##         BB             0.000 0.490 0.000
## AB      AA             0.210 0.210 0.000
##         AB             0.105 0.210 0.105
##         BB             0.000 0.210 0.210
## BB      AA             0.000 0.090 0.000
##         AB             0.000 0.045 0.045
##         BB             0.000 0.000 0.090
```

Collect evidence



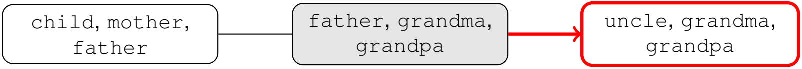

```
s2_old <- s2
s2 <- tabMarg(q3, c("grandma", "grandpa")); ftable(s2)
```

```
##         grandpa   AA   AB   BB
## grandma
## AA             0.49 0.42 0.09
## AB             0.49 0.42 0.09
## BB             0.49 0.42 0.09
```

```
q2 <- tabMult(tabDiv(s2, s2_old), q2); ftable(q2)
```

```
##                 father     AA     AB     BB
## grandma grandpa
## AA      AA             0.2401 0.0000 0.0000
##         AB             0.1029 0.1029 0.0000
##         BB             0.0000 0.0441 0.0000
## AB      AA             0.1029 0.1029 0.0000
##         AB             0.0441 0.0882 0.0441
##         BB             0.0000 0.0189 0.0189
## BB      AA             0.0000 0.0441 0.0000
##         AB             0.0000 0.0189 0.0189
##         BB             0.0000 0.0000 0.0081
```

Distribute evidence



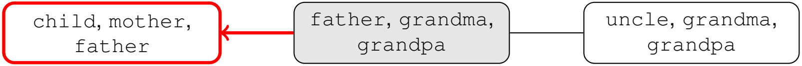

```
s2_old <- s2
s2 <- tabMarg(q2, c("grandma", "grandpa")); ftable(s2)
```

```
##          grandpa     AA     AB     BB
## grandma
## AA               0.2401 0.2058 0.0441
## AB               0.2058 0.1764 0.0378
## BB               0.0441 0.0378 0.0081
```

```
q3 <- tabMult(tabDiv(s2, s2_old), q3); ftable(q3)
```

```
##                 uncle     AA     AB     BB
## grandpa grandma
## AA      AA            0.2401 0.0000 0.0000
##         AB            0.1029 0.1029 0.0000
##         BB            0.0000 0.0441 0.0000
## AB      AA            0.1029 0.1029 0.0000
##         AB            0.0441 0.0882 0.0441
##         BB            0.0000 0.0189 0.0189
## BB      AA            0.0000 0.0441 0.0000
##         AB            0.0000 0.0189 0.0189
##         BB            0.0000 0.0000 0.0081
```

Distribute evidence




```
s1_old <- s1
s1 <- tabMarg(q2, "father"); s1
```

```
## father
##   AA   AB   BB
## 0.49 0.42 0.09
```

```
q1 <- tabMult(tabDiv(s1, s1_old), q1); ftable(q1)
```

```
##              father     AA     AB     BB
## child mother
## AA    AA            0.2401 0.1029 0.0000
##       AB            0.1029 0.0441 0.0000
##       BB            0.0000 0.0000 0.0000
## AB    AA            0.0000 0.1029 0.0441
##       AB            0.1029 0.0882 0.0189
##       BB            0.0441 0.0189 0.0000
## BB    AA            0.0000 0.0000 0.0000
##       AB            0.0000 0.0441 0.0189
##       BB            0.0000 0.0189 0.0081
```

Potentials are now marginal distributions

```
ftable(tabMarg(joint, c("father", "mother", "child")))
```

```
##                child     AA     AB     BB
## father mother                            
## AA     AA             0.2401 0.0000 0.0000
##        AB             0.1029 0.1029 0.0000
##        BB             0.0000 0.0441 0.0000
## AB     AA             0.1029 0.1029 0.0000
##        AB             0.0441 0.0882 0.0441
##        BB             0.0000 0.0189 0.0189
## BB     AA             0.0000 0.0441 0.0000
##        AB             0.0000 0.0189 0.0189
##        BB             0.0000 0.0000 0.0081
```

```
ftable(q1, row.vars=c("father", "mother"))
```

```
##                child     AA     AB     BB
## father mother                            
## AA     AA             0.2401 0.0000 0.0000
##        AB             0.1029 0.1029 0.0000
##        BB             0.0000 0.0441 0.0000
## AB     AA             0.1029 0.1029 0.0000
##        AB             0.0441 0.0882 0.0441
##        BB             0.0000 0.0189 0.0189
## BB     AA             0.0000 0.0441 0.0000
##        AB             0.0000 0.0189 0.0189
##        BB             0.0000 0.0000 0.0081
```

Example: propagation of evidence

The same junction tree is used for the representation of various conditional distributions – we just modify the potentials by *entering evidence*.

When entering evidence `father = AA`, we

- Choose a potential containing the variable `father`
- Set the value of the potential to 0 for all combinations except where `father == AA`.
- The product of potentials is now $p(\texttt{family})\mathbb{1}_{\{\texttt{father == AA}\}}$ rather than $p(\texttt{family})$.
- propagation gives the (unnormalised) marginal potentials for the conditional distribution given the evidence.

Enter the evidence

```
## Modify the potential for a clique containing father
q1 <- setSliceValue(q1, list(father = "AA"), comp=T)
ftable(q1, col.vars="father")

##                 father      AA     AB     BB
## child mother
## AA    AA                0.2401 0.0000 0.0000
##       AB                0.1029 0.0000 0.0000
##       BB                0.0000 0.0000 0.0000
## AB    AA                0.0000 0.0000 0.0000
##       AB                0.1029 0.0000 0.0000
##       BB                0.0441 0.0000 0.0000
## BB    AA                0.0000 0.0000 0.0000
##       AB                0.0000 0.0000 0.0000
##       BB                0.0000 0.0000 0.0000
```

Propagate the evidence

```
## Run propagation again
s1_old <- s1
s1 <- tabMarg(q1, "father")
q2 <- tabMult(tabDiv(s1, s1_old), q2)
s2_old <- s2
s2 <- tabMarg(q3, c("grandma", "grandpa"))
q2 <- tabMult(tabDiv(s2, s2_old), q2)
s2_old <- s2
s2 <- tabMarg(q2, c("grandma", "grandpa"))
q3 <- tabMult(tabDiv(s2, s2_old), q3)
s1_old <- s1
s1 <- tabMarg(q2, "father")
q1 <- tabMult(tabDiv(s1, s1_old), q1)
```

Potentials are now marginals

```
net_with_ev <- setEvidence(extended.family, "father", "AA")
marg <- querygrain(net_with_ev, c("father", "grandma", "grandpa"), type = "joint")
ftable(marg, row.vars=c("grandma", "grandpa"))
```

```
##                   father   AA   AB   BB
## grandma grandpa
## AA      AA               0.49 0.00 0.00
##         AB               0.21 0.00 0.00
##         BB               0.00 0.00 0.00
## AB      AA               0.21 0.00 0.00
##         AB               0.09 0.00 0.00
##         BB               0.00 0.00 0.00
## BB      AA               0.00 0.00 0.00
##         AB               0.00 0.00 0.00
##         BB               0.00 0.00 0.00
```

```
ftable(q2, row.vars=c("grandma", "grandpa"))
```

```
##                   father     AA     AB     BB
## grandma grandpa
## AA      AA               0.2401 0.0000 0.0000
##         AB               0.1029 0.0000 0.0000
##         BB               0.0000 0.0000 0.0000
## AB      AA               0.1029 0.0000 0.0000
##         AB               0.0441 0.0000 0.0000
##         BB               0.0000 0.0000 0.0000
## BB      AA               0.0000 0.0000 0.0000
##         AB               0.0000 0.0000 0.0000
##         BB               0.0000 0.0000 0.0000
```

Normalise potentials

Because we started with an unnormalised probability mass function, we need to normalise the potentials after propagation.

The normalising constant is exactly the probability of the evidence.

```
sum(q2)
```

```
## [1] 0.49
```

```
pEvidence(net_with_ev)
```

```
## [1] 0.49
```

The junction tree representation

The junction tree representation allows local computations on batches of variables: we never need to compute the joint distribution!

In particular, we can easily

- marginalize
- get conditional distributions
- simulate

`gRain` has two convenience functions for setting up the junction tree:

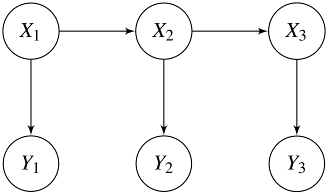
`compile`: sets up the junction tree and initialises potentials

`propagate`: modifies the potentials into marginal distributions

These steps are done implicitly when `querygrain` is called, so it is more efficient to do it explicitly.

Including non-discrete observed variables

Example: Hidden Markov Models



Discrete unobservable variables $X_1 \sim \text{bin}(2, 1/3)$ and $X_i \,|\, X_{i-1} \sim \text{bin}(2, X_{i-1}/3)$.

Continuous observable variables Y_i with $Y_i \,|\, X_i \sim N(X_i, 1)$.

Build the network for X

```r
## HMM network
x1 <- parray("x1", list(0:2), dbinom(0:2, 2, 1/3))
x2 <- parray(c("x2", "x1"), list(0:2, 0:2),
              outer(0:2, 0:2, function(x,y)dbinom(x, 2, y/3)))
x3 <- parray(c("x3", "x2"), list(0:2, 0:2),
              outer(0:2, 0:2, function(x,y)dbinom(x, 2, y/3)))
hmm <- grain(compileCPT(list(x1, x2, x3)))
plot(hmm)
hmm <- propagate(hmm, compile = TRUE)
```

Enter observations Y via likelihood evidence

Observations $y = (0, 4, 1)$ are entered into the network via *likelihood evidence*:

```
## The evidence is the vector p(y_i | x_i) for all
## possible values of x_i
(evidence <- list(x1 = dnorm(0, mean = 0:2, sd = 1),
                  x2 = dnorm(4, mean = 0:2, sd = 1),
                  x3 = dnorm(1, mean = 0:2, sd = 1)))
```

```
## $x1
## [1] 0.39894228 0.24197072 0.05399097
##
## $x2
## [1] 0.0001338302 0.0044318484 0.0539909665
##
## $x3
## [1] 0.2419707 0.3989423 0.2419707
```

```
hmm_ev <- setEvidence(hmm, evidence = evidence)
hmm_ev <- propagate(hmm_ev)
```

Posterior distributions and likelihood

After propagating the likelihood evidence the network represents $p(x \mid y)$.

```
## probability p(x2, x3 | y)
querygrain(hmm_ev, c("x2", "x3"), type = "joint")
```

```
##    x3
## x2          0         1          2
##   0 0.02263374 0.0000000 0.00000000
##   1 0.07445755 0.1227597 0.01861439
##   2 0.06567849 0.4331421 0.26271397
```

We can get the likelihood $p(y)$ as the normalising constant

```
## p(y)
pEvidence(hmm_ev)
```

```
## [1] 0.0003230195
```

How does it work?

Setting likelihood evidence for a variable corresponds to multiplying one potential – and thus the entire distribution $p(x_1, x_2, x_3)$ – by the likelihood evidence:

$$p(x_1, x_2, x_3 \mid y_1, y_2, y_3) \propto p(x_1, x_2, x_3) p(y_1 \mid x_1) p(y_2 \mid x_2) p(y_3 \mid x_3)$$

The normalising constant (`pEvidence`) is exactly the likelihood of the observations:

$$p(y_1, y_2, y_3) = \sum_{x_1, x_2, x_3} p(x_1, x_2, x_3) p(y_1 \mid x_1) p(y_2 \mid x_2) p(y_3 \mid x_3)$$