Graphical Models and Bayesian Networks

Tutorial
International Workshop on Statistical Modelling

Rennes, France, 2016

Søren Højsgaard

Department of Mathematical Sciences

Aalborg University, Denmark

July 5, 2016

Printed: July 5, 2016 File: bayesnet-slides.tex

Contents

1 Outline **5**
- 1.1 Book: Graphical Models with R . . . 6
- 1.2 Package versions . . . 7

2 Example: The sprinkler network **8**
- 2.1 The setting . . . 9
- 2.2 Conditional probability tables (CPTs) . . . 12
- 2.3 A small digression: Operations on arrays . . . 14
- 2.4 Using Bayes' formula . . . 17

3 The curse of dimensionality **19**

4 Conditional independence **20**

5 Example: Mendelian segregation **23**
- 5.1 Mendel's Genetics . . . 24
- 5.2 Now with two children . . . 35
- 5.3 Exercises . . . 37
- 5.4 Building a network . . . 38
- 5.5 Joint/marginal distributions . . . 40
- 5.6 Evidence . . . 41
- 5.7 Probability of configuration of set of variables . . . 43
- 5.8 Simulation . . . 44
- 5.9 Example: Paternity testing . . . 45

6 Missing father, but the uncle is available **47**
- 6.1 Exercises . . . 50

7 Example: The chest clinic narrative **51**
- 7.1 DAG–based models . . . 53
- 7.2 Conditional probability tables (CPTs) . . . 54

8 An introduction to the gRain package **56**
8.1 Specify BN from list of CPTs . . . 57
8.2 Specify BN from DAG and data . . . 61
8.3 Querying the network . . . 63
8.4 Setting evidence . . . 64

9 The curse of dimensionality **68**
9.1 So what is the problem? . . . 71

10 Example: Lizard data **72**
10.1 Conditional independence and data . . . 73
10.2 DAG factorization . . . 75
10.3 Extracting CPTs . . . 76

11 Behind the scenes **78**
11.1 Message passing in the lizard example . . . 79
11.2 How to - in R . . . 83
11.3 Collect Evidence . . . 85
11.4 How to – in R . . . 86
11.5 Distribute Evidence . . . 87
11.6 How to – in R . . . 88
11.7 It works - empirical proof . . . 89
11.8 Setting evidence . . . 93
11.9 How to – in R . . . 94

12 Learning – from data to BNs **98**

13 Conditional independence – recap **100**

14 Discrete data and contingency tables **102**
14.1 Extracting CPTs . . . 103
14.2 Creating BN from CPTs . . . 106

15 Log–linear models **107**
15.1 Log–linear models and conditional independence 109
15.2 How to - in R 110
15.3 Dependence graphs 112
15.4 Decomposable log–linear models 114

16 Testing for conditional independence **115**
16.1 What is a CI-test – stratification 116

17 Log–linear models with gRim **119**

18 The reinis data **124**

19 From graphical model to BN **128**

20 The coronary artery disease data **129**

21 From graph and data to network **135**

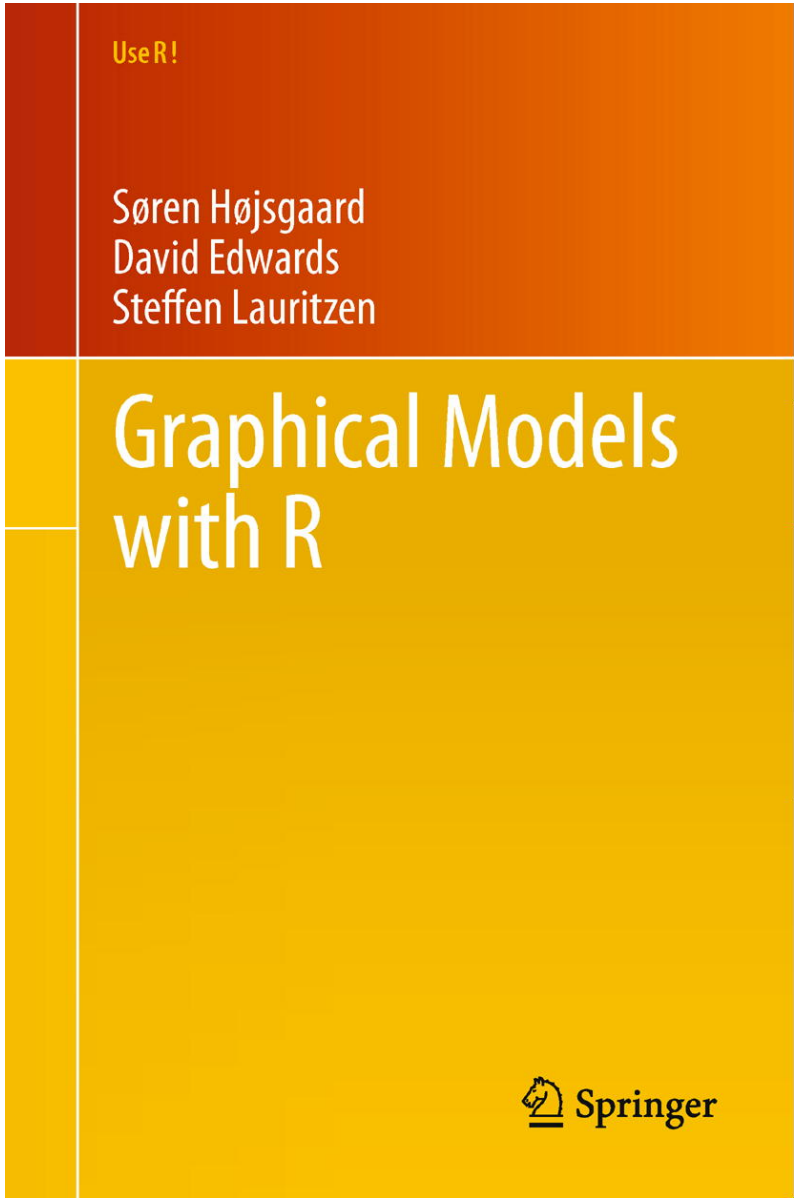
22 Prediction **136**

23 Winding up **140**

1 Outline

- Bayesian networks and the **gRain** package
- Probability propagation; conditional independence restrictions and dependency graphs
- Learning structure with log–linear, graphical and decomposable models for contingency tables
- Using the **gRim** package for structural learning.
- Convert decomposable model to Bayesian network.
- Other packages for structure learning.

1.1 Book: Graphical Models with R



1.2 Package versions

We shall in this tutorial use the R–packages **gRbase**, **gRain** and **gRim**.

Installation: First install bioconductor packages

```
> source("http://bioconductor.org/biocLite.R");
> biocLite(c("graph","RBGL","Rgraphviz"))
```

Then install **gRbase**, **gRain** and **gRim** from CRAN

```
> install.packages("gRbase", dependencies=TRUE)
> install.packages("gRain", dependencies=TRUE)
> install.packages("gRim", dependencies=TRUE)
```

Go to `http://people.math.aau.dk/~sorenh/software/gR` and locate the development versions (on which this tutorial is based).

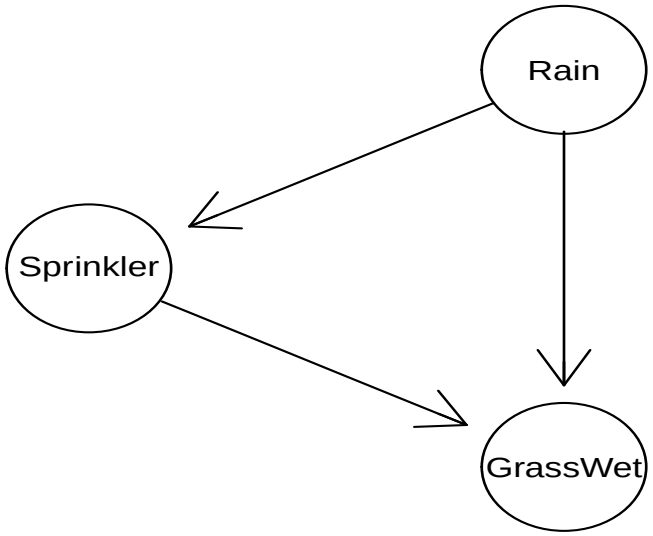
```
> packageVersion("gRbase")
[1] '1.7.6'
> packageVersion("gRain")
[1] '1.2.6'
> packageVersion("gRim")
[1] '0.1.17'
```

2 Example: The sprinkler network

2.1 The setting

Two events can cause grass to be wet: Either the sprinkler is on or it is raining. Rain has a direct effect on the use of the sprinkler: when it rains, the sprinkler is usually not turned on.

What is the probability that it has been raining given that the grass is wet?



This can be modeled with a Bayesian network. The variables (R)ain, (S)prinkler, (G)rassWet have two possible values: (y)es and (n)o.

Using Bayes' formula twice, a joint probability mass function (pmf) $p_{GSR}(g, s, r)$ can be factorized as

$$p_{GSR}(g, s, r) = p_{G|SR}(g|s, r)p_{S|R}(s|r)p_R(r)$$

We shall allow the more informal notation

$$P(G, S, R) = P(G|S, R)P(S, R) = P(G|S, R)P(S|R)P(R)$$

In a model building context we start in "the other end" by specifying the terms

$$P(G|S, R), \quad P(S|R), \quad P(R)$$

We call these terms conditional probability tables (or CPTs).

Then we construct a joint pmf by

$$P(G, S, R) \leftarrow P(G|S, R)P(S|R)P(R)$$

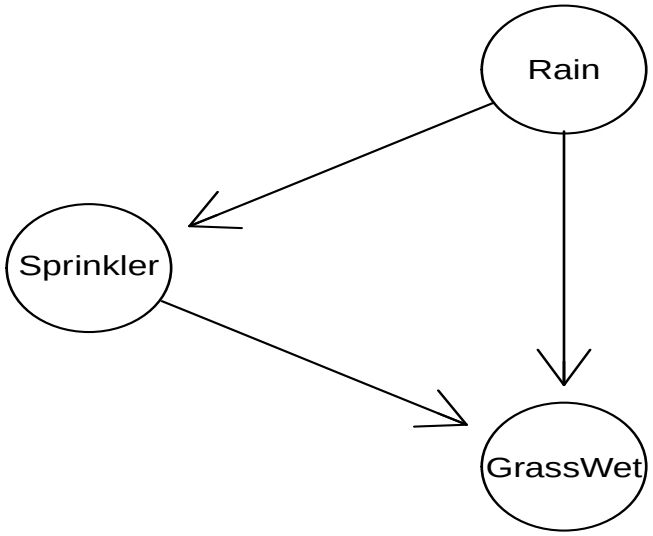
Notice this: Terms on the right hand of

$$P(G, S, R) = P(G|S, R)P(S|R)P(R)$$

has the form

$$p(v|parent(v))$$

relative to the DAG (directed acyclic graph):



2.2 Conditional probability tables (CPTs)

For compact printing of arrays define utility function

```
> flat <- function(x){ftable(x, row.vars=1)} ## Just a utility
```

Conditional probability tables (CPTs) in R are arrays.

Arrays can be created e.g. with `array()` or as follows:

```
> yn <- c("yes", "no")
> uni <- list(Rain = yn, Sprinkler = yn, GrassWet = yn)
> ## P(R)
> p.R <- arMk("Rain", levels=uni, values=c(.2, .8))
> p.R
Rain
yes  no
0.2 0.8
> ## P(S|R)
> p.S_R <- arMk(c("Sprinkler","Rain"), levels = uni,
                values=c(.01, .99, .4, .6))
> p.S_R %>% flat
          Rain  yes   no
Sprinkler
yes             0.01 0.40
no              0.99 0.60
```

```
> ## P(G|S,R)
> p.G_SR <- arMk(~GrassWet:Sprinkler:Rain, levels = uni,
                 values=c(.99, .01, .8, .2, .9, .1, 0, 1))
> p.G_SR  %>% flat
          Sprinkler  yes         no
          Rain       yes   no  yes   no
GrassWet
yes                  0.99 0.90 0.80 0.00
no                   0.01 0.10 0.20 1.00
```

With Rs `array()` function we can do the same as:

```
> yn <- c("yes", "no")
> p.R <- array( c(.2, .8), dim = 2,
                dimnames = list(Rain=yn) )
> p.S_R <- array(c(.01, .99, .4, .6), dim=c(2, 2),
                 dimnames=list(Sprinkler=yn, Rain=yn))
> p.G_SR <- array(c(.99, .01, .8, .2, .9, .1, 0, 1),
                  dim = c(2, 2, 2),
                  dimnames=list(GrassWet=yn, Sprinkler=yn, Rain=yn))
```

2.3 A small digression: Operations on arrays

```
> T1 <- arMk( ~a:b, levels=c(2,2), values=1:4 )
> T2 <- arMk( ~b:c, levels=c(2,2), values=5:8 )
> T1 %>% flat
   b b1 b2
a
a1    1  3
a2    2  4
> T2 %>% flat
   c c1 c2
b
b1    5  7
b2    6  8
```

Think of T_1 as function of variables (a, b) and T_2 as function of (b, c).

The product $T = T_1 T_2$ is a function of (a, b, c) defined as

$$T(a, b, c) \leftarrow T_1(a, b) T_2(b, c)$$

```
> T1 %>% flat
   b b1 b2
a
a1    1  3
a2    2  4
> T2 %>% flat
   c c1 c2
b
b1    5  7
b2    6  8
> arprod( T1, T2 ) %>% flat
   c c1    c2
   a a1 a2 a1 a2
b
b1    5 10  7 14
b2   18 24 24 32
> # or arMult( T1, T2 )
> # or T1 %a*% T2
```

Joint pmf:

```
> ## P(G,S,R)
> p.GSR <- arprod( p.G_SR, p.S_R, p.R )
> p.GSR  %>% flat
     Sprinkler     yes              no
     GrassWet      yes      no      yes      no
Rain
yes             0.00198 0.00002 0.15840 0.03960
no              0.28800 0.03200 0.00000 0.48000
> sum( p.GSR ) # check
[1] 1
```

2.4 Using Bayes' formula

Question: What is the probability that it is raining given that the grass is wet?

Answer: Use Bayes formula:

$$P(R|G=y) = \frac{P(R,G=y)}{P(G=y)} = \frac{\sum_{S=y,n} P(R,S,G=y)}{\sum_{R=y,n;S=y,n} P(R,S,G=y)}$$

This question - and others - can be answered as:

```
> arDist(p.GSR, marg="Rain", cond=list(GrassWet="yes"))
Rain
  yes    no
0.358 0.642
> arDist(p.GSR, cond=list(GrassWet="yes"))
     Sprinkler
Rain      yes    no
  yes 0.00442 0.353
  no  0.64231 0.000
```

In detail we have:

```
> ## Marginalize -> P(R,G)
> p.RG  <- arMarg(p.GSR, c("Rain", "GrassWet")); p.RG
     GrassWet
Rain     yes     no
  yes 0.160 0.0396
  no  0.288 0.5120
> ## Marginalize -> P(G)
> p.G   <- arMarg(p.RG, "GrassWet"); p.G
GrassWet
  yes    no
0.448 0.552
> ## Condition -> P(R|G)
> p.R_G <- arDiv(p.RG, p.G); p.R_G
         Rain
GrassWet    yes    no
     yes 0.3577 0.642
     no  0.0718 0.928
> ## Pick the slice -> P(R|G=yes)
> arSlice( p.R_G, slice=list(GrassWet="yes"))
  yes    no
0.358 0.642
```

3 The curse of dimensionality

In the example, the joint state space is $2^3 = 8$.

We calculated the joint pmf (a $2 \times 2 \times 2$ table) by multiplying conditionals, then we marginalized and then we conditioned.

With 80 variables each with 10 levels, the joint state space is $10^{80} \approx$ the number of atoms in the universe.

Fortunately, there is often a structure to the model such that one need NOT calculate the full joint pmf.

Instead we do local computations on low dimensional tables and "send messages" between them.

This structure arise from conditional independence restrictions.

gRain exploits this structure and has been used succesfully on networks with 10.000s of variables.

4 Conditional independence

Conditional independence restrictions are essential in Bayesian networks and graphical models.

Let X, Y, Z be random variables.

The statement that X and Z are conditionally independent given Y, written $X \perp\!\!\!\perp Z|Y$, means that X and Z are independent in the conditional distribution given given $Y = y$ for each possible value y of Y.

In terms of a joint density we have

$$f(x, z|y) = f(x|y)f(z|y)$$

or equivalently that

$$f(x|y, z) = f(x|y)$$

Once we know y we will obtain no additional information about x by also getting to know z.

Independence is a "special case" of conditional independence where we need not "condition on anything": $X \perp\!\!\!\perp Y$ iff

$$f(x, y) = f(x)f(y)$$

or - equivalently -

$$f(x \,|\, y) = f(x)$$

In practice it is often easiest to check conditional independence using the factorization criterion: If

$$f(x, y, z) = q_1(x, y)q_2(y, z)$$

for non–negative functions $q_1()$ and $q_2()$ then $X \perp\!\!\!\perp Z|Y$.

Example 4.1 *Example:* $(X_1, X_2, X_3) \sim N_3(\mu, \Sigma)$ *with*

$$\Sigma^{-1} = K = \begin{bmatrix} k_{11} & k_{12} & 0 \\ k_{21} & k_{22} & k_{23} \\ 0 & l_{32} & k_{33} \end{bmatrix}$$

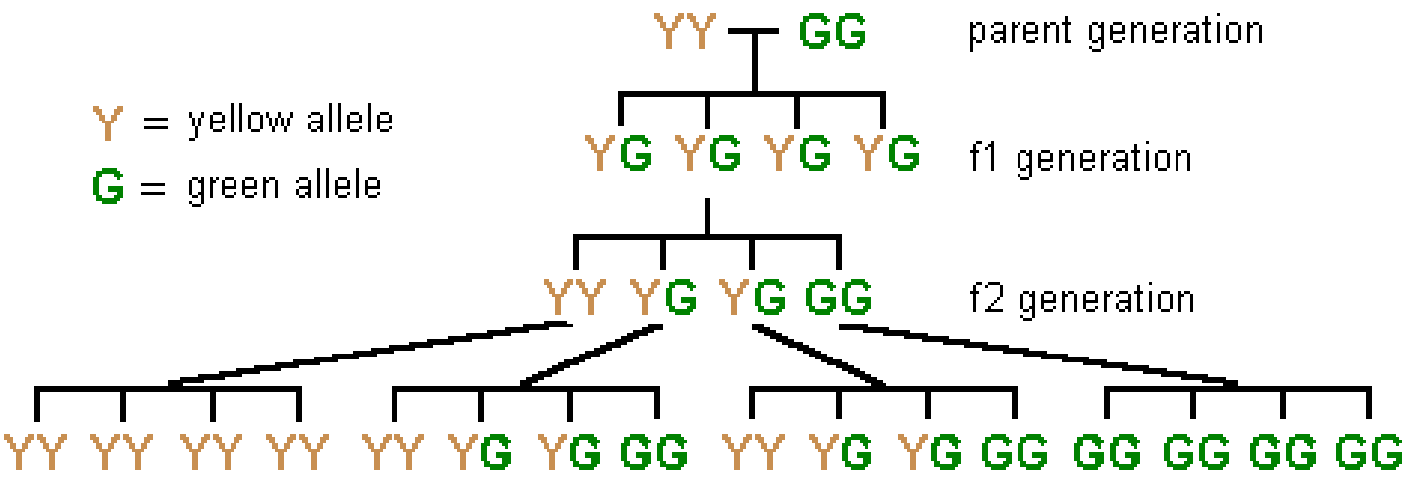
We have $X_1 \perp\!\!\!\perp X_3 \mid$ *All other variables , i.e.* $X_1 \perp\!\!\!\perp X_3 \mid X_2$. □

5 Example: Mendelian segregation

5.1 Mendel's Genetics

A very clear introduction at:

`http://anthro.palomar.edu/mendel/mendel_1.htm`



A pea will have two alleles related to its color. A pea recieves one allele from each parent. An allele can be y or g.

The genotype is an unordered pair of alleles: $\{y, y\}$, $\{y, g\}$ or $\{g, g\}$. Think of genotype as a random variable with states $\{yy, yg, gg\}$.

```
> gts <- c("yy", "yg", "gg")
[1] "yy" "yg" "gg"
```

The alleles combine independently and therefore the probability of each genotype is

```
> gtprobs <- c(.25, .50, .25)
[1] 0.25 0.50 0.25
```

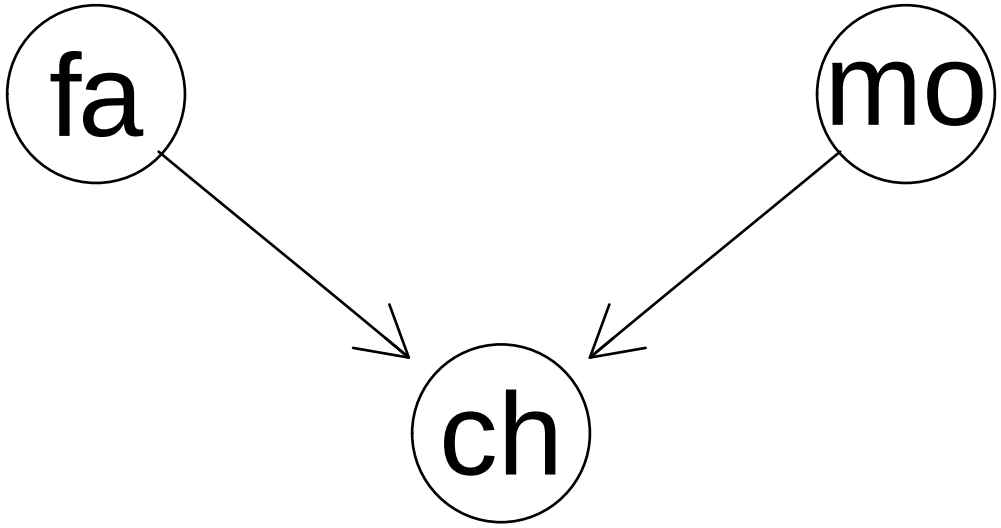
The phenotype is what can be observed, i.e. wheter the pea is yellow or green.

The y–allele is dominant; the g–allele is recessive: The pea will be green (the phenotype) only if the genotype is gg the pea will be green; if the allele is yy or yg, the pea will be yellow.

Therefore yellow and gree peas will appear in the proportions $3 : 1$.

Peas do not have fathers and mothers, but only parents - but let us for later purposes think of them as fathers and mothers.

```
> dG <- dag(~ch|fa:mo)
> plot( dG )
```



```
> men <- mendel( c("y","g"), names = c("ch", "fa", "mo") )
> head( men, 4 )
  ch fa mo prob
1 yy yy yy  1.0
2 yg yy yy  0.0
3 gg yy yy  0.0
4 yy yg yy  0.5
> dim( men )
[1] 27  4
> ## For later use, save inheritance probabilities
> inheritance <- men$prob
> head( inheritance )
[1] 1.0 0.0 0.0 0.5 0.5 0.0
```

Conditional distribution $p(ch \mid fa, mo)$:

```
> ssp <- list(fa = gts, mo = gts, ch = gts)
> p.c_fm <- arMk(~ch:fa:mo, levels=ssp, values=inheritance)
> ftable( p.c_fm, row.vars = "ch" )
   fa   yy             yg             gg
   mo   yy   yg   gg   yy   yg   gg   yy   yg   gg
ch
yy    1.00 0.50 0.00 0.50 0.25 0.00 0.00 0.00 0.00
yg    0.00 0.50 1.00 0.50 0.50 0.50 1.00 0.50 0.00
gg    0.00 0.00 0.00 0.00 0.25 0.50 0.00 0.50 1.00
```

In equilibrium the population frequencies of the alleles will be

```
> gts
[1] "yy" "yg" "gg"
> gtprobs
[1] 0.25 0.50 0.25
> p.fa <- arMk(~fa, levels=ssp, values=gtprobs)
> p.fa
fa
  yy   yg   gg
0.25 0.50 0.25
> p.mo <- arMk(~mo, levels=ssp, values=gtprobs)
> p.mo
mo
  yy   yg   gg
0.25 0.50 0.25
>
```

Joint distribution is $p(ch, fa, mo) = (ch|fa, mo)p(fa)p(mo)$:

```
> joint <- arprod( p.fa, p.mo, p.c_fm )
> ftable(round( joint, 3) , row.vars="ch")
   fa    yy                  yg                  gg
   mo    yy    yg    gg      yy    yg    gg      yy    yg    gg
ch
yy      0.062 0.062 0.000   0.062 0.062 0.000   0.000 0.000 0.000
yg      0.000 0.062 0.062   0.062 0.125 0.062   0.062 0.062 0.000
gg      0.000 0.000 0.000   0.000 0.062 0.062   0.000 0.062 0.062
```

Marginal distributions:

```
> ## Not surprising:
> arDist( joint, marg="fa")
fa
  yy   yg   gg
0.25 0.50 0.25
> ## Because of equilibrium
> arDist( joint, marg="ch")
ch
  yy   yg   gg
0.25 0.50 0.25
```

Now condition on mother:

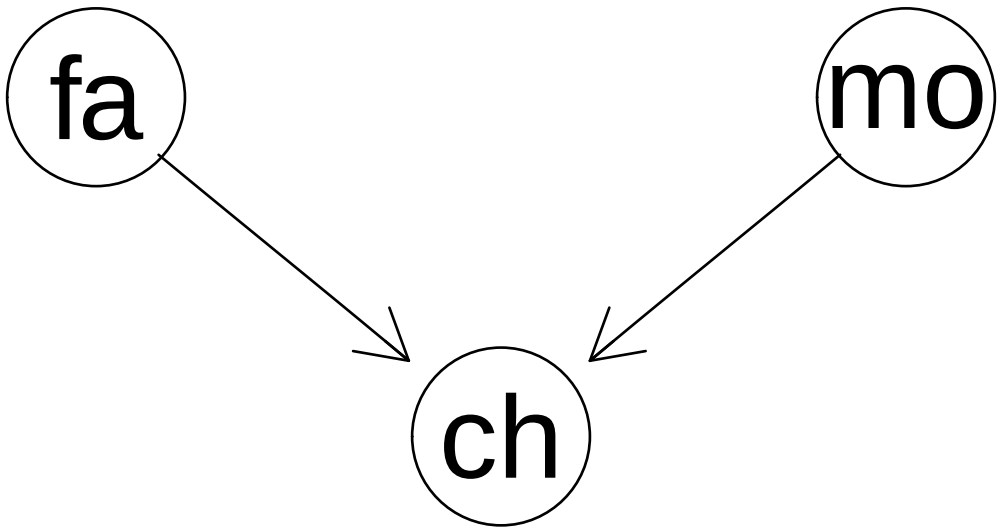
```
> arDist( joint, marg="fa", cond=list(mo="yy"))
fa
  yy   yg   gg
0.25 0.50 0.25
> arDist( joint, marg="ch", cond=list(mo="yy"))
ch
 yy  yg  gg
0.5 0.5 0.0
```

Conclusions?

Now condition on child

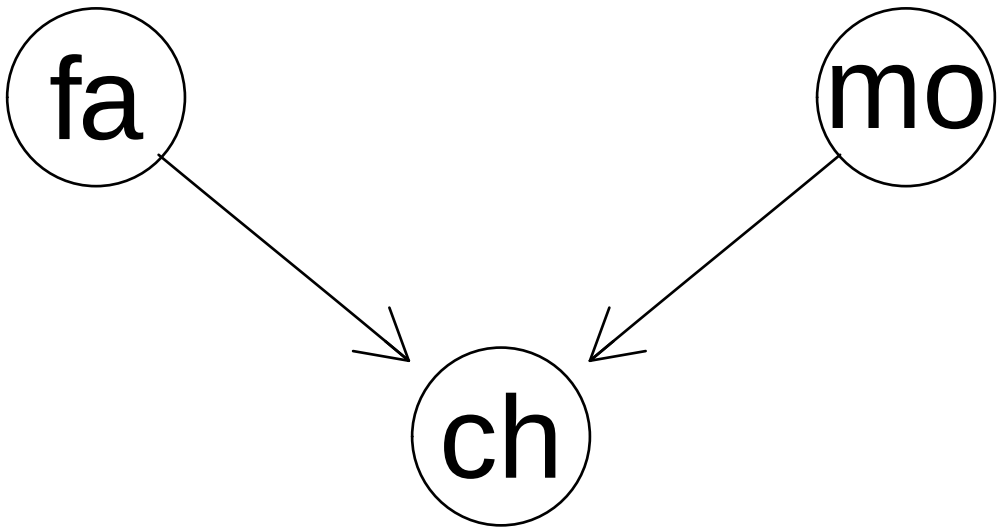
```
> arDist(joint, marg="fa")
fa
  yy   yg   gg
0.25 0.50 0.25
> arDist(joint, marg="fa", cond=list(ch="gg"))
fa
 yy  yg  gg
0.0 0.5 0.5
> arDist(joint, marg="fa", cond=list(ch="yg"))
fa
  yy   yg   gg
0.25 0.50 0.25
> arDist(joint, marg="fa", cond=list(ch="yg", mo="gg"))
fa
 yy  yg  gg
0.5 0.5 0.0
```

Conclusions?



Marginally, fa and mo are independent:

$$p(fa, mo) = \sum_{ch}(ch|fa, mo)p(fa)p(mo) = p(fa)p(mo)$$

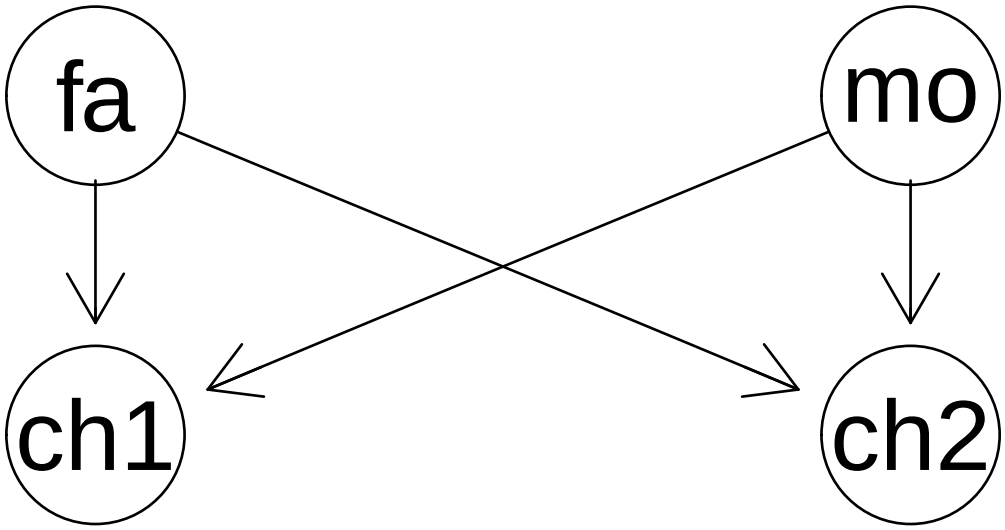


Joint distribution is $p(ch, fa, mo) = p(ch|fa, mo)p(fa)p(mo)$.

But conditionally on ch, fa and mo are NOT independent: We do not have a factorization:

$$p(ch, fa, mo) = (ch|fa, mo)p(fa)p(mo) \neq g(ch, fa)h(ch, mo)$$

5.2 Now with two children



Joint distribution is:

$$p(ch2, ch2, fa, mo) = p(ch1|fa, mo)p(ch2|fa, mo)p(fa)p(mo)$$

Clearly $ch1 \perp\!\!\!\perp ch2 \,|\, (fa, mo)$ because

$$p(ch2, ch2, fa, mo) = g(ch1, fa, mo)h(ch2, fa, mo)$$

```
> ssp <- list(fa = gts, mo = gts, ch = gts, ch1 = gts, ch2 = gts)
> head( inheritance )
[1] 1.0 0.0 0.0 0.5 0.5 0.0
> p.c1_fm <- arMk(~ch1:fa:mo, levels=ssp, values=inheritance)
> p.c2_fm <- arMk(~ch2:fa:mo, levels=ssp, values=inheritance)
> joint <- arprod(p.fa, p.mo, p.c1_fm, p.c2_fm)
> joint  %>% round(3) %>% ftable(col.vars = c("ch1", "ch2"))
      ch1    yy                   yg                   gg
      ch2    yy    yg    gg    yy    yg    gg    yy    yg    gg
fa mo
yy yy     0.062 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
   yg     0.031 0.031 0.000 0.031 0.031 0.000 0.000 0.000 0.000
   gg     0.000 0.000 0.000 0.000 0.062 0.000 0.000 0.000 0.000
yg yy     0.031 0.031 0.000 0.031 0.031 0.000 0.000 0.000 0.000
   yg     0.016 0.031 0.016 0.031 0.062 0.031 0.016 0.031 0.016
   gg     0.000 0.000 0.000 0.000 0.031 0.031 0.000 0.031 0.031
gg yy     0.000 0.000 0.000 0.000 0.062 0.000 0.000 0.000 0.000
   yg     0.000 0.000 0.000 0.000 0.031 0.031 0.000 0.031 0.031
   gg     0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.062
```

5.3 Exercises

1. On your own computer, construct the 4–way table above.
2. What is $p(ch1|ch2 = yy)$?
3. What is $p(ch1|ch2 = yy, fa = yy)$?
4. What is $p(ch1|ch2 = yy, fa = yy, mo = yg)$?
5. What is $p(ch1|fa = yy, mo = yg)$?

Hint: Your friend is `arDist`

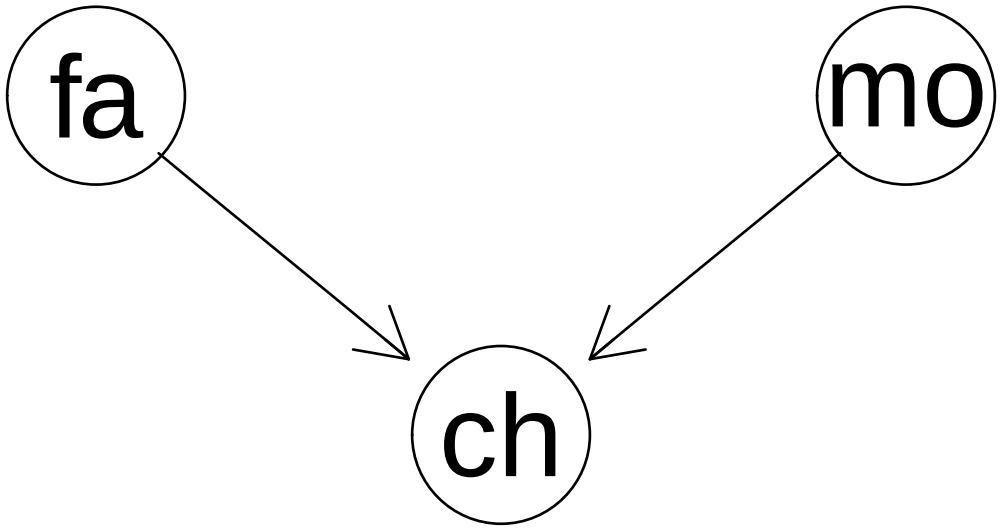
5.4 Building a network

We have seen things done by brute force computations; now we build at bayesian network:

For later purposes we shall assume that the genotype frequencies are not the Mendelian ones but:

```
> gtprobs2 <- c(.09, .42, .49)
> gts
[1] "yy" "yg" "gg"
> str( ssp )
List of 5
 $ fa : chr [1:3] "yy" "yg" "gg"
 $ mo : chr [1:3] "yy" "yg" "gg"
 $ ch : chr [1:3] "yy" "yg" "gg"
 $ ch1: chr [1:3] "yy" "yg" "gg"
 $ ch2: chr [1:3] "yy" "yg" "gg"
```

```
> p.fa <- arMk(~fa, levels=ssp, values=gtprobs2)
> p.mo <- arMk(~mo, levels=ssp, values=gtprobs2)
> cptl <- compileCPT( list( p.fa, p.mo, p.c_fm ) ); cptl
CPTspec with probabilities:
 P( fa )
 P( mo )
 P( ch | fa mo )
> trio <- grain( cptl )
> trio
Independence network: Compiled: FALSE Propagated: FALSE
  Nodes: chr [1:3] "fa" "mo" "ch"
> plot( trio )
```



5.5 Joint/marginal distributions

```
> qgrain( trio, nodes = c("fa", "ch"),
          type="marginal" ) ## Default type
$fa
fa
  yy   yg   gg
0.09 0.42 0.49

$ch
ch
  yy   yg   gg
0.09 0.42 0.49
> qgrain( trio, nodes = c("fa", "ch"), type="joint")  %>%
      ftable(row.vars="ch")
   fa    yy    yg    gg
ch
yy     0.027 0.063 0.000
yg     0.063 0.210 0.147
gg     0.000 0.147 0.343
```

5.6 Evidence

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

1. conditional distribution given the evidence
2. marginal probability of the evidence

```
> ## Network with evidence entered
> trio_ev <- setEvidence(trio, nodes=c("ch", "mo"),
                           states = c("yg", "yy"))
> ## p(father | child = yg, mother = yy)
> qgrain(trio_ev, nodes="fa")
$fa
fa
 yy  yg  gg
0.0 0.3 0.7
> ## Removing all entered evidence
> trio_ev2 <- retractEvidence(trio_ev)
> ## p(father)
> qgrain(trio_ev2, nodes="fa")
$fa
fa
  yy   yg   gg
0.09 0.42 0.49
```

5.7 Probability of configuration of set of variables

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

```
> qgrain(trio, nodes=c("ch", "mo"), type = "joint")
    ch
mo       yy    yg    gg
  yy 0.027 0.063 0.000
  yg 0.063 0.210 0.147
  gg 0.000 0.147 0.343
```

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

```
> tr <- setEvidence(trio, evidence = c(ch="yg", mo="yy"))
> pEvidence(tr)
[1] 0.063
```

5.8 Simulation

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

```
> ## Prior distribution
> simulate(trio, 4)
  fa mo ch
1 gg gg gg
2 gg gg gg
3 yg yg gg
4 yy gg yg
> ## Posterior after observing child and mother
> simulate(trio_ev, 4)
  fa mo ch
1 gg yy yg
2 yg yy yg
3 gg yy yg
4 gg yy yg
```

5.9 Example: Paternity testing

A "mother pea" with genotype yy has a child with genotype yg.

She claims that "Pea X", who has genotype yg, 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: Pea X is the father

vs.

H_2: Some unknown pea is the father

We need to compute

$$LR = \frac{Pr(ch = yg, m = yy \,|\, H_1)}{Pr(ch = yg, m = yy \,|\, H_2)} = \frac{Pr(ch = yg, m = yy \,|\, fa = yg)}{Pr(ch = yg, m = yy)}$$

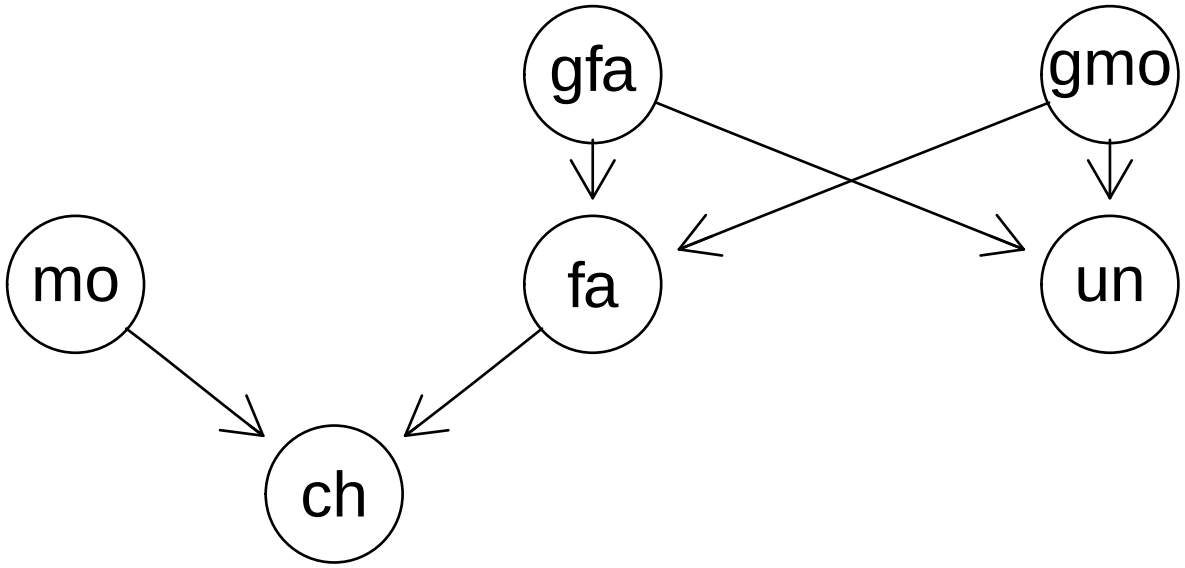
$$LR = \frac{Pr(ch = yg, m = yy \mid fa = yg)}{Pr(ch = yg, m = yy)}$$

```
> ## P(m = yy, c = yg, f = yg)
> p.fmc <- pEvidence(setEvidence(trio,
                                 evidence = list(mo = "yy",
                                                 ch = "yg", fa = "yg")
> ## P(f = yg)
> p.f <- pEvidence(setEvidence(trio,
                               evidence = list( fa = "yg")))
> L.H1 <- p.fmc/p.f
> ## P(m = yy, c = yg)
> L.H2 <- pEvidence(setEvidence(trio,
                                evidence = list( mo = "yy", ch = "yg")
> ## Likelihood ratio comparing "Pea X" vs unknown pea.
> L.H1/L.H2
[1] 0.714
```

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

6 Missing father, but the uncle is available

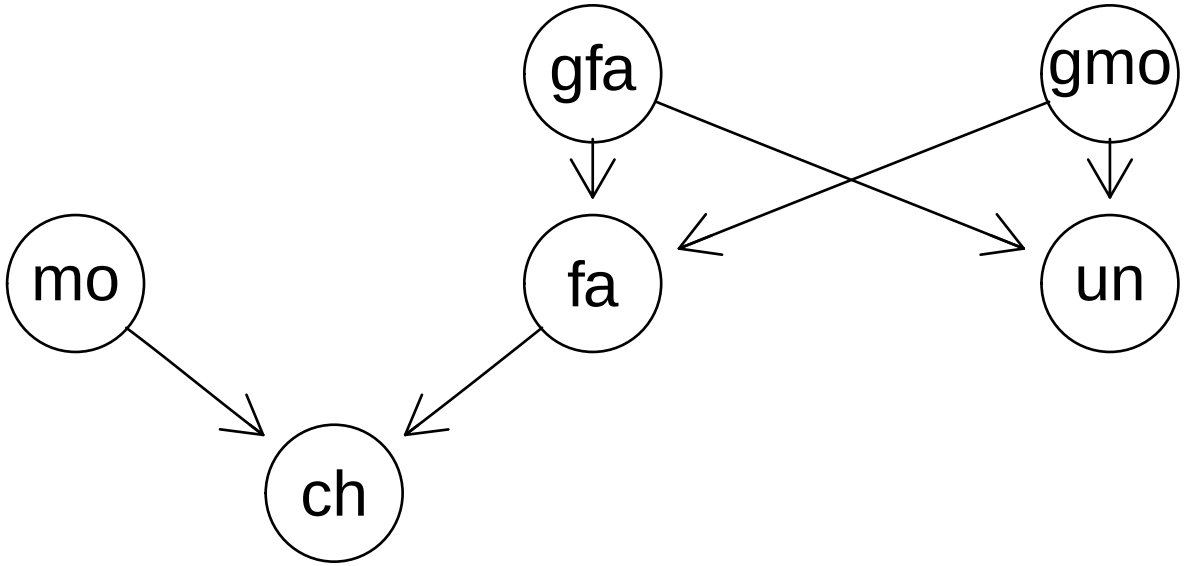
```
> dG2 <- dag(~ch|mo:fa + fa|gfa:gmo + un|gfa:gmo)
> plot(dG2)
```



$$p(c, m, f, gf, gm, u) \quad = p(c|m, f)p(f|gf, gm)p(u|gf, gm)p(m)p(gf)p(gm)$$

```
> ssp <- list(fa = gts, mo = gts, ch = gts, ch1 = gts, ch2 = gts,
              gfa = gts, gmo = gts, un = gts)
> ## p(m), p(gm), p(gf)
> p.m  <- arMk(~mo, levels=ssp, values=gtprobs2)
> p.gm <- arMk(~gmo, levels=ssp, values=gtprobs2)
> p.gf <- arMk(~gfa, levels=ssp, values=gtprobs2)
> ## p(child | mother, father)
> p.c_fm <- arMk(~ch:mo:fa, levels=ssp, values=inheritance)
> ## p(father | grandma, grandpa)
> p.f_gfgm <- arMk(~fa:gfa:gmo, levels=ssp, values=inheritance)
> ## p(uncle | grandma, grandpa)
> p.u_gfgm <- arMk(~un:gfa:gmo, levels=ssp, values=inheritance)
> c.mf <- parray( c("child", "mother", "father"),
  levels = rep(list(gts), 3),
  values = inheritance)
> cpt.list <- compileCPT(list(p.c_fm, p.m, p.f_gfgm,
                              p.u_gfgm, p.gm, p.gf))
> extended.family <- grain(cpt.list)
```

```
> extended.family
Independence network: Compiled: FALSE Propagated: FALSE
  Nodes: chr [1:6] "ch" "mo" "fa" "un" "gmo" "gfa"
> plot(extended.family)
```



6.1 Exercises

1. Build the network `extended.family` on your own computer.

2. A mother pea claims that Pea X is the father of her child pea. Unfortunately it is not possible to get a DNA sample from Pea X, but his brother ("uncle") is willing to give a sample.

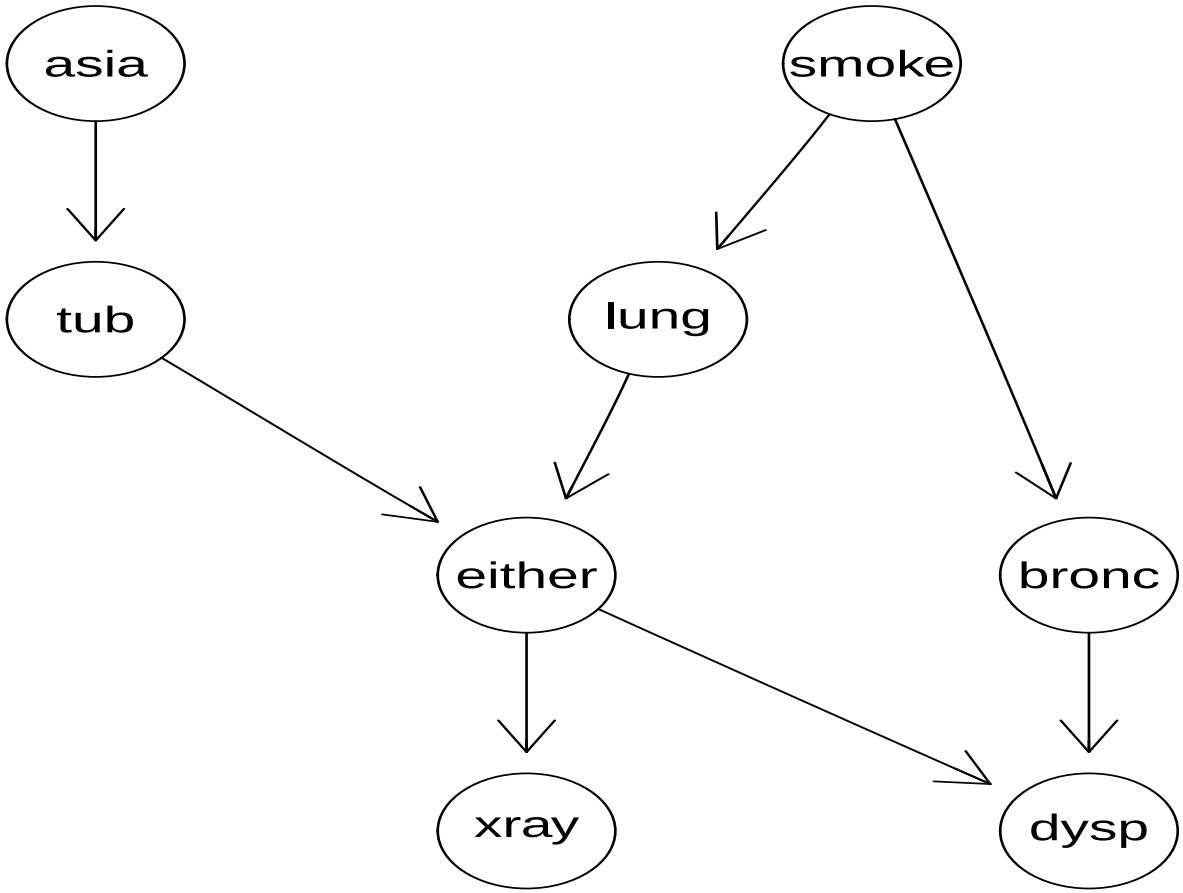
mother	yg
child	yg
uncle	gg

   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 yg?

7 Example: The chest clinic narrative

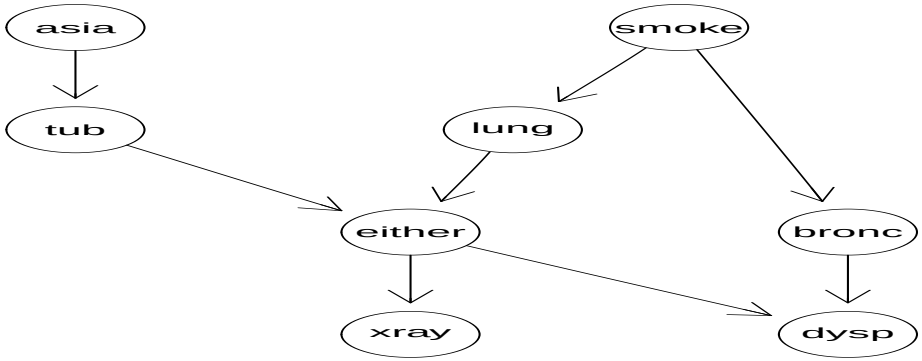


Lauritzen and Spiegehalter (1988) present the following narrative:

- "Shortness–of–breath (*dyspnoea*) may be due to *tuberculosis*, *lung cancer* or *bronchitis*, or none of them, or more than one of them.
- A recent visit to *Asia* increases the chances of tuberculosis, while *smoking* is known to be a risk factor for both lung cancer and bronchitis.
- The results of a single chest *X–ray* do not discriminate between lung cancer and tuberculosis, as n*either* does the presence or absence of dyspnoea."

The narrative can be pictured as a DAG (Directed Acyclic Graph)

7.1 DAG–based models



With an informal notation, a joint distribution for all variables

$$\begin{aligned} V &= \{Asia, Tub, Smoke, Lung, Either, Bronc, Xray, Dysp\} \\ &\equiv \{a, t, s, l, e, b, x, d\} \end{aligned}$$

can be obtained as

$$p(V) = \prod_v p(v|pa(v))$$

which here boils down to

$$p(V) = p(a)p(t|a)p(s)p(l|s)p(b|s)p(e|t,l)p(d|e,b)p(x|e).$$

7.2 Conditional probability tables (CPTs)

In R, CPTs are just multiway arrays WITH dimnames attribute. For example $p(t|a)$:

```
> yn <- c("yes","no");
> x <- c(5,95,1,99)
> # Vanilla R
> t.a <- array(x, dim=c(2,2), dimnames=list(tub=yn,asia=yn))
> t.a
     asia
tub   yes no
  yes   5  1
  no   95 99
> # Alternative specification: arMk() from gRbase
> uni <- list(asia=yn, tub=yn)
> t.a <- arMk(~tub:asia, levels=uni, values=x)
> t.a
     asia
tub   yes no
  yes   5  1
  no   95 99
```

```
> # Alternative (partial) specification
> t.a  <- cptable(~tub | asia, values=c(5,95,1,99), levels=yn)
> t.a
{v,pa(v)} : chr [1:2] "tub" "asia"
    <NA> <NA>
yes    5    1
no    95   99
```

Last case: Only names of v and $pa(v)$ and levels of v are definite; the rest is inferred in the context; see later.

8 An introduction to the **gRain** package

8.1 Specify BN from list of CPTs

Specify chest clinic network. Can be done in many ways; one is from a list of CPTs:

```
> yn <- c("yes","no")
> a    <- cptable(~asia,          values=c(1,99), levels=yn)
> t.a  <- cptable(~tub | asia,    values=c(5,95,1,99), levels=yn)
> s    <- cptable(~smoke,         values=c(5,5), levels=yn)
> l.s  <- cptable(~lung | smoke,  values=c(1,9,1,99), levels=yn)
> b.s  <- cptable(~bronc | smoke, values=c(6,4,3,7), levels=yn)
> e.lt <- cptable(~either | lung:tub,
                  values=c(1,0,1,0,1,0,0,1), levels=yn)
> x.e  <- cptable(~xray | either,
                  values=c(98,2,5,95), levels=yn)
> d.be <- cptable(~dysp | bronc:either,
                  values=c(9,1,7,3,8,2,1,9), levels=yn)
```

```
> cpt.list <- compileCPT(list(a, t.a, s, l.s, b.s, e.lt, x.e, d.be))
> cpt.list
CPTspec with probabilities:
 P( asia )
 P( tub | asia )
 P( smoke )
 P( lung | smoke )
 P( bronc | smoke )
 P( either | lung tub )
 P( xray | either )
 P( dysp | bronc either )
```

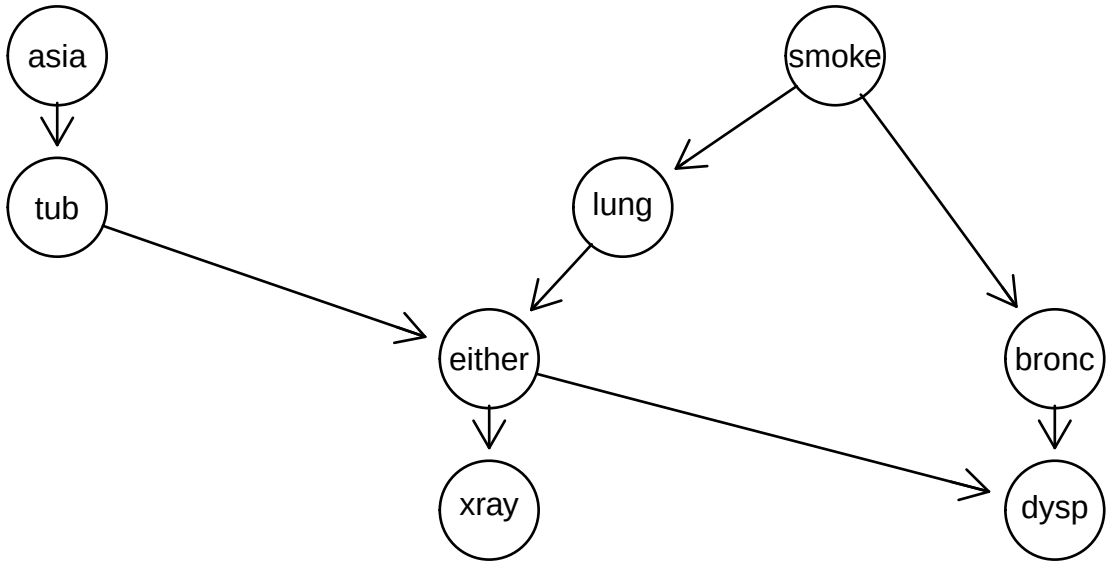
```
> cpt.list$asia
asia
 yes   no
0.01 0.99
> cpt.list$tub
     asia
tub    yes   no
  yes 0.05 0.01
  no  0.95 0.99
> ftable(cpt.list$either, row.vars=1) # Notice: logical variable
       lung yes     no
       tub  yes no yes no
either
yes             1   1   1  0
no              0   0   0  1
```

```
> # Create network from CPT list:
> bn  <- grain(cpt.list)
> # Compile network (details follow)
> bn <- compile(bn)
> bn
Independence network: Compiled: TRUE Propagated: FALSE
  Nodes: chr [1:8] "asia" "tub" "smoke" "lung" "bronc" ...
```

8.2 Specify BN from DAG and data

If the structure of the DAG is known and we have data, we can just do:

```
> vpa <- list("asia", c("tub", "asia"), "smoke", c("lung", "smoke"),
              c("bronc","smoke"), c("either", "lung", "tub"),
              c("xray", "either"), c("dysp", "bronc", "either"))
> dg <- dag( vpa )
> plot(dg)
```



```
> data(chestSim1000, package="gRbase")
> head(chestSim1000)
  asia tub smoke lung bronc either xray dysp
1   no  no    no   no   yes     no   no  yes
2   no  no   yes   no   yes     no   no  yes
3   no  no   yes   no    no     no   no   no
4   no  no    no   no    no     no   no   no
5   no  no   yes   no   yes     no   no  yes
6   no  no   yes  yes   yes    yes  yes  yes
> bn2 <- grain(dg, data=chestSim1000, smooth=.1)
> bn2
Independence network: Compiled: FALSE Propagated: FALSE
  Nodes: chr [1:8] "asia" "tub" "smoke" "lung" "bronc" ...
```

The CPTs are estimated as the relative frequencies.

8.3 Querying the network

```
> # Query network to find marginal probabilities of diseases
> disease <- c("tub","lung","bronc")
> querygrain(bn, nodes=disease)
$tub
tub
   yes     no
0.0104 0.9896

$lung
lung
  yes    no
0.055 0.945

$bronc
bronc
 yes   no
0.45 0.55
```

8.4 Setting evidence

```
> # Set evidence and query network again
> bn.ev <- setEvidence(bn, evidence=list(asia="yes",dysp="yes"))
> querygrain(bn.ev, nodes=disease)
$tub
tub
   yes     no
0.0878 0.9122

$lung
lung
   yes     no
0.0995 0.9005

$bronc
bronc
  yes    no
0.811 0.189
```

```
> # Get the evidence
> getEvidence(bn.ev)
Univariate evidence
  nodes is.hard.evidence hard.state
1  asia              TRUE        yes
2  dysp              TRUE        yes
[[1]]
asia
yes  no
  1   0

[[2]]
dysp
yes  no
  1   0
> # Probability of observing the evidence (the normalizing constant)
> pEvidence(bn.ev)
[1] 0.0045
```

```
> # Set additional evidence and query again
> bn.ev <- setEvidence(bn.ev, evidence=list(xray="yes"))
> querygrain(bn.ev, nodes=disease)
$tub
tub
  yes    no
0.392 0.608

$lung
lung
  yes    no
0.444 0.556

$bronc
bronc
  yes    no
0.629 0.371
```

```
> # Get joint dist of tub, lung, bronc given evidence
> x <- querygrain(bn.ev, nodes=disease, type="joint")
> ftable( x, row.vars=1 )
    lung       yes                 no
    bronc      yes       no        yes       no
tub
yes         0.01406 0.00816 0.18676 0.18274
no          0.26708 0.15497 0.16092 0.02531
> bn.ev <- retractEvidence(bn.ev, nodes="asia")
> bn.ev
Independence network: Compiled: TRUE Propagated: TRUE
  Nodes: chr [1:8] "asia" "tub" "smoke" "lung" "bronc" ...
  Evidence:
  nodes is.hard.evidence hard.state
1  dysp              TRUE        yes
2  xray              TRUE        yes
  pEvidence: 0.070670
```

A little shortcut: Most uses of **gRain** involves 1) setting evidence into a network and 2) querying nodes. This can be done in one step:

```
> querygrain(bn, evidence=list(asia="yes",dysp="yes"),
                       nodes=disease)
$tub
tub
   yes     no
0.0878 0.9122

$lung
lung
   yes     no
0.0995 0.9005

$bronc
bronc
  yes    no
0.811 0.189
```

9 The curse of dimensionality

In principle (and in practice in this small toy example) we can find e.g. $p(b|a^+, d^+)$ by brute force calculations.

Recall: We have a collection of conditional probability tables (CPTs) of the form $p(v|pa(v))$:

$$\{p(a), p(t|a), p(s), p(l|s), p(b|s), p(e|t,l), p(d|e,b), p(x|e)\}$$

Brute force computations:

1) Form the joint distribution $p(V)$ by multiplying the CPTs

$$p(V) = p(a)p(t|a)p(s)p(l|s)p(b|s)p(e|t,l)p(d|e,b)p(x|e).$$

This gives $p(V)$ represented by a table with giving a table with $2^8 = 256$ entries.

2) Find the marginal distribution $p(a, b, d)$ by marginalizing $p(V) = p(a, t, s, k, e, b, x, d)$

$$p(a, b, d) = \sum_{t,s,k,e,b,x} p(t, s, k, e, b, x, d)$$

This is table with $2^3 = 8$ entries.

3) Lastly notice that $p(b|a^+, d^+) \propto p(a^+, b, d^+)$.

Hence from $p(a, b, d)$ we must extract those entries consistent with $a = a^+$ and $d = d^+$ and normalize the result.

Alternatively (and easier): Set all entries not consistent with $a = a^+$ and $d = d^+$ in $p(a, b, d)$ equal to zero.

9.1 So what is the problem?

In chest clinic example the joint state space is $2^8 = 256$.

With 80 variables each with 10 levels, the joint state space is $10^{80} \approx$ the number of atoms in the universe!

Still, **gRain** has been succesfully used in a genetics network with 80.000 nodes... How can this happen?

The trick is to NOT to calculate the joint distribution

$$p(V) = p(a)p(t|a)p(s)p(l|s)p(b|s)p(e|t,l)p(d|e,b)p(x|e).$$

explicitly because that leads to working with high dimensional tables.

Instead we do local computations on on low dimensional tables and "send messages" between them.

The challenge is to organize these local computations.

10 Example: Lizard data

Characteristics of 409 lizards were recorded, namely species (S), perch diameter (D) and perch height (H).

```
> data(lizardRAW, package="gRbase")
> head(lizardRAW, 4)
  diam height species
1   >4  >4.75    dist
2   >4  >4.75    dist
3  <=4 <=4.75   anoli
4   >4 <=4.75   anoli
```

Defines 3–way contingency table.

```
> data(lizard, package="gRbase")
> ftable( lizard, row.vars=1 )
     height  >4.75       <=4.75
     species anoli dist  anoli dist
diam
<=4             32   61     86   73
>4              11   41     35   70
```

10.1 Conditional independence and data

Conditional independependence $height \perp\!\!\!\perp diam | species$ (short: $h \perp\!\!\!\perp d \mid s$) means independence between `height` and `diam` in each `species`–slice

```
> lizard
, , species = anoli

     height
diam  >4.75 <=4.75
  <=4    32     86
  >4     11     35

, , species = dist

     height
diam  >4.75 <=4.75
  <=4    61     73
  >4     41     70
```

Seems reasonable!

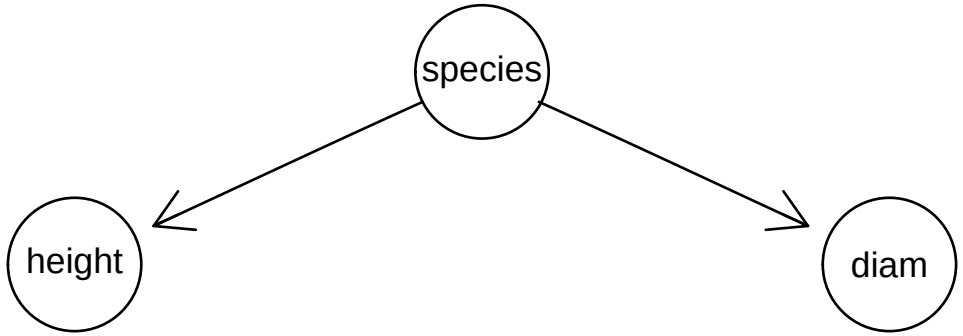
Tests for independence in slices support this:

```
> chisq.test( lizard[, , 1] )
        Pearson's Chi-squared test with Yates' continuity correction

data:  lizard[, , 1]
X-squared = 0.05, df = 1, p-value = 0.8
> chisq.test( lizard[, , 2] )
        Pearson's Chi-squared test with Yates' continuity correction

data:  lizard[, , 2]
X-squared = 2, df = 1, p-value = 0.2
```

10.2 DAG factorization

A DAG that encodes $d \perp\!\!\!\perp h \mid s$ is

```
> d <- dag( ~species + height|species + diam|species ); plot(d)
```



Joint distribution

$$p(d, h, s) = p(h|s)p(d|s)p(s)$$

The general picture: A missing edge implies a (conditional) independence: $p(d, h, s) = q_1(d, s)q_2(h, s)$.

More generally: Factorization according to DAG:

$$p(V) = \prod_v p(v \mid pa(v))$$

10.3 Extracting CPTs

```
> ## Extract empirical distributions
> s   <- arMarg(lizard, ~species); s
species
anoli  dist
  164   245
> h_s <- arMarg(lizard, ~height + species); h_s
         species
height    anoli dist
  >4.75      43  102
  <=4.75    121  143
> d_s <- arMarg(lizard, ~diam + species); d_s
      species
diam   anoli dist
  <=4    118  134
  >4      46  111
```

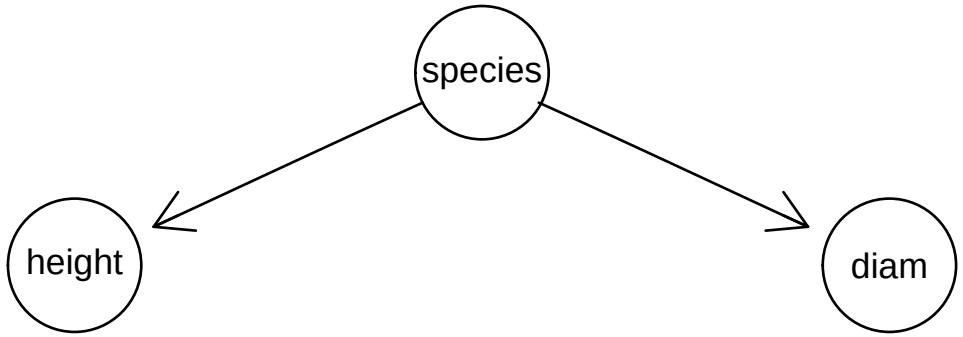
```
> ## Normalize to CPTs if desired (not necessary because
> ## we can always normalize at the end)
> p.s   <- arNormalize(s, "first"); p.s
species
anoli  dist
0.401 0.599
> p.h_s <- arNormalize(h_s, "first"); p.h_s
        species
height   anoli  dist
  >4.75  0.262 0.416
  <=4.75 0.738 0.584
> p.d_s <- arNormalize(d_s, "first"); p.d_s
     species
diam  anoli  dist
  <=4  0.72 0.547
  >4   0.28 0.453
```

We can multiply, marginalize and condition as we did before.

11 Behind the scenes

11.1 Message passing in the lizard example

```
> d <- dag( ~species + height|species + diam|species ); plot(d)
```



Joint distribution has the form

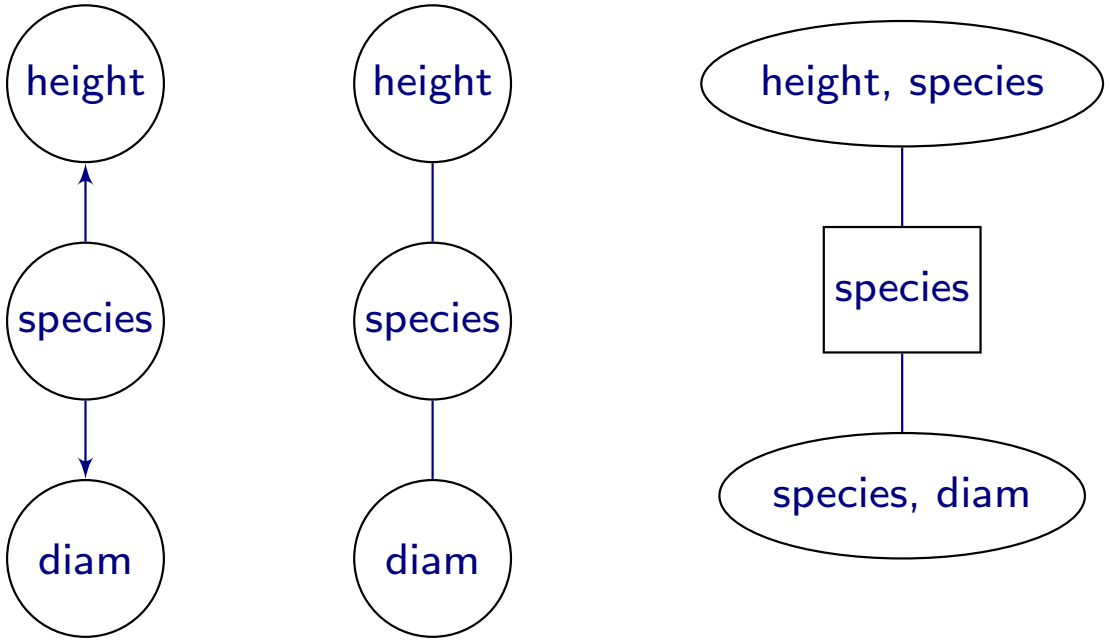
$$p(d, h, s) = p(h|s)p(d|s)p(s)$$

Terms on right hand side are given, and we can – in principle – multiply these to produce the joint distribution. (In practice we can do so in this low–dimensional case (a 2^3 table).)

However, we want to avoid forming the joint distribution and still be able to compute e.g. $p(h)$, $p(h|d)$ or $p(h|d, s)$.

From now on we no longer need the DAG. Instead we use an undirected graph to dictate the message passing:

The "moral graph" is obtained by 1) marrying parents and 2) dropping directions. The moral graph is (in this case) triangulated which means that the cliques can be organized in a tree called a junction tree.



Define $q_1(d,s) = p(s)p(d|s)$ and $q_2(h,s) = p(h|s)$ and we have

$$p(d,h,s) = p(s)p(d|s)p(h|s) = q_1(d,s)q_2(h,s)$$

The q–functions are defined on the cliques of the moral graph or - equivalently - on the nodes of the junction tree.

The q–functions are called potentials; they are non–negative functions but they are typically not probabilities and they are hence difficult to interpret. Think of q–functions as interactions.

The factorization

$$p(d, h, s) = q_1(d, s)q_2(h, s)$$

is called a clique potential representation.

Goal: We shall operate on q–functions such that at the end they will contain the marginal distributions, i.e.

$$q_1(d, s) = p(d, s), \quad q_2(h, s) = p(h, s)$$

11.2 How to - in R

Before we extracted the CPTs from data

```
> list(p.s, p.d_s, p.h_s)
[[1]]
species
anoli  dist
0.401 0.599

[[2]]
     species
diam  anoli  dist
  <=4  0.72 0.547
  >4   0.28 0.453

[[3]]
        species
height   anoli  dist
  >4.75  0.262 0.416
  <=4.75 0.738 0.584
```

Define q_1 and q_2:

```
> q1.ds <- arprod(p.s, p.d_s); q1.ds
```

```
      species
diam   anoli  dist
  <=4 0.289 0.328
  >4  0.112 0.271
> q2.hs <- p.h_s; q2.hs
         species
height    anoli  dist
  >4.75  0.262 0.416
  <=4.75 0.738 0.584
```

11.3 Collect Evidence

Pick any node, say $(height, species) = (h, s)$ as root in the junction tree, and work inwards towards the root as follows.

First, define $q_1(s) \leftarrow \sum_d q_1(d, s)$.

We have

$$p(d, h, s) = q_1(d, s)q_2(h, s) = \left[\frac{q_1(d, s)}{q_1(s)}\right]\left[q_2(h, s)q_1(s)\right]$$

Therefore, we update potentials as

$$q_1(d, s) \leftarrow q_1(d, s)/q_1(s), \quad q_2(h, s) \leftarrow q_2(h, s)q_1(s)$$

and the new potentials are also defined on the nodes of the junction tree. We still have

$$p(d, h, s) = q_1(d, s)q_2(h, s)$$

11.4 How to – in R

Updating of potentials is done as follows:

```
> q1.s  <- arMarg(q1.ds, "species"); q1.s
species
anoli  dist
0.401 0.599
> q2.hs <- arMult(q2.hs, q1.s); q2.hs
        height
species >4.75 <=4.75
  anoli 0.105  0.296
  dist  0.249  0.350
> q1.ds <- arDiv(q1.ds, q1.s); q1.ds
        diam
species   <=4    >4
  anoli 0.720 0.280
  dist  0.547 0.453
```

11.5 Distribute Evidence

Next work outwards from the root (h, s).

Set $q_2(s) \leftarrow \sum_h q_2(h, s)$. We have

$$p(d, h, s) = q_1(d, s)q_2(h, s) = \frac{\big[q_1(d, s)q_2(s)\big]\, q_2(h, s)}{q_2(s)}$$

We therefore update as $q_1(d, s) \leftarrow q_1(d, s)q_2(s)$ and have

$$p(d, h, s) = q_1(d, s)q_2(h, s) = \frac{q_1(d, s)q_2(h, s)}{q_2(s)}$$

The form above is called the clique marginal representation and the main point is that q_1 and q_2 "fit on their marginals", i.e. $q_1(s) = q_2(s)$ and

$$q_1(d, s) = p(d, s), \quad q_2(h, s) = p(h, s)$$

11.6 How to – in R

```
> q2.s  <- arMarg(q2.hs, "species"); q2.s
species
anoli  dist
0.401 0.599
> q1.ds <- arMult(q1.ds, q2.s); q1.ds
        diam
species   <=4    >4
  anoli 0.289 0.112
  dist  0.328 0.271
```

11.7 It works - empirical proof

The joint distribution $p(d, h, s)$ is a $2 \times 2 \times 2$ array which we really do not want to calculate this in general; here we just do it as "proof of concept":

```
> p.dhs <- arprod( p.s, p.d_s, p.h_s )
> ftable( p.dhs, row.vars="species")
        height  >4.75         <=4.75
        diam       <=4     >4    <=4     >4
species
anoli           0.0756 0.0295 0.2129 0.0830
dist            0.1364 0.1130 0.1912 0.1584
```

Claim: After these steps $q_1(d, s) = p(d, s)$ and $q_2(h, s) = p(h, s)$. That is, we have the marginal distributions on the cliques:

Proof:

```
> q1.ds
        diam
species   <=4    >4
  anoli 0.289 0.112
  dist  0.328 0.271
> arDist(p.dhs, ~diam+species)
     species
diam  anoli  dist
  <=4 0.289 0.328
  >4  0.112 0.271
> q2.hs
        height
species >4.75 <=4.75
  anoli 0.105  0.296
  dist  0.249  0.350
> arDist(p.dhs, ~height+species)
         species
height    anoli  dist
  >4.75  0.105 0.249
```

<=4.75 0.296 0.350

Now we can obtain, e.g. $p(h)$ as

```
> p.h <- arDist(q2.hs, "height")
> p.h
height
 >4.75 <=4.75
 0.355  0.645
```

And we NEVER calculated the full joint distribution!

11.8 Setting evidence

Next consider the case where we have the evidence that `diam>4`.

```
> q1.ds <- arSliceMult(q1.ds, list(diam=">4"))
> q1.ds
        diam
species <=4    >4
  anoli   0 0.112
  dist    0 0.271
```

11.9 How to – in R

```
> # Repeat all the same steps as before
> q1.s  <- arMarg(q1.ds, "species")
> q2.hs <- arMult(q2.hs, q1.s)
> q1.ds <- arDiv(q1.ds, q1.s)
> q2.s  <- arMarg(q2.hs, "species")
> q1.ds <- arMult(q1.ds, q2.s)
```

Claim: After these steps $q_1(d,s) = p(d,s|d^+)$ and $q_2(h,s) = p(h,s|d^+)$.

```
> #joint <- arSliceMult(p.dhs, list(diam=">4"), comp=T);
> joint <- arSliceMult(p.dhs, list(diam=">4"))
> ftable( joint, row.vars=1)
       species  anoli           dist
       diam       <=4     >4    <=4     >4
height
>4.75          0.0000 0.0295 0.0000 0.1130
<=4.75         0.0000 0.0830 0.0000 0.1584
```

Proof:

```
> q1.ds ## Needs normalization
        diam
species <=4    >4
  anoli   0 0.112
  dist    0 0.271
> q1.ds / sum( q1.ds )
        diam
species <=4    >4
  anoli   0 0.293
  dist    0 0.707
> arDist(joint, ~diam:species)
     species
diam  anoli  dist
  <=4 0.000 0.000
  >4  0.293 0.707
```

```
> q2.hs ## Needs normalization
        height
species >4.75 <=4.75
  anoli 0.105  0.296
  dist  0.249  0.350
> q2.hs / sum( q2.hs )
        height
species >4.75 <=4.75
  anoli 0.105  0.296
  dist  0.249  0.350
> arDist(joint, ~height:species)
         species
height     anoli  dist
  >4.75  0.0768 0.294
  <=4.75 0.2162 0.413
```

And we NEVER calculated the full joint distribution!

12 Learning – from data to BNs

- If we know the DAG, we can estimate the CPTs from data as relative frequencies.
- If we don't know the DAG then we can find a DAG from data using statistical methods.
- From the perspection of computations in the network, the DAG is not used. All computations are based on a junction tree.
- If we know a junction tree we can estimate clique marginals from data.
- If we do not know either, we can find a junction tree from data using statistical methods.
- One way ahead: A junction tree corresponds to a special type of statistical model: A decomposable graphical model.
- We shall use data for finding a decomposable graphical model, and then convert this to a BN.

13 Conditional independence – recap

Conditional independence restrictions are essential in Bayesian networks and graphical models.

Let X, Y, Z be random variables.

The statement that X and Z are conditionally independent given Y, written $X \perp\!\!\!\perp Z|Y$, means that X and Z are independent in the conditional distribution given given $Y = y$ for each possible value y of Y.

In terms of a joint density we have

$$f(x, z|y) = f(x|y)f(z|y)$$

or equivalently that

$$f(x|y, z) = f(x|y)$$

Once we know y we will obtain no additional information about x by also getting to know z.

In practice it is often easiest to check conditional independence using the

factorization criterion: If

$$f(x, y, z) = q_1(x, y)q_2(y, z)$$

for non–negative functions $q_1()$ and $q_2()$ then $X \perp\!\!\!\perp Z|Y$.

14 Discrete data and contingency tables

Characteristics of 409 lizards were recorded, namely species (S), perch diameter (D) and perch height (H).

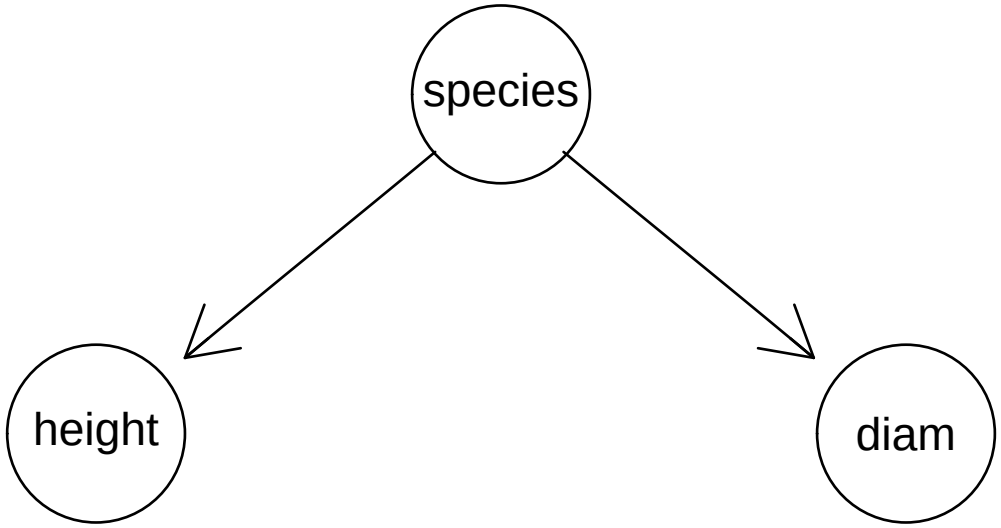
```
> data(lizardRAW, package="gRbase")
> head(lizardRAW, 4)
  diam height species
1   >4  >4.75    dist
2   >4  >4.75    dist
3  <=4 <=4.75   anoli
4   >4 <=4.75   anoli
```

Defines 3–way contingency table.

```
> data(lizard, package="gRbase")
> ftable( lizard, row.vars=1 )
     height  >4.75       <=4.75
     species anoli dist  anoli dist
diam
<=4             32   61     86   73
>4              11   41     35   70
```

14.1 Extracting CPTs

```
> d <-dag( ~species + height|species + diam|species ); plot(d)
```



Joint distribution

$$p(s, b, d) = p(h|s)p(d|s)p(s)$$

```
> ## Extract empirical distributions
> s   <- tabMarg(lizard, ~species); s
species
anoli  dist
  164   245
> h_s <- tabMarg(lizard, ~height + species); h_s
        species
height   anoli dist
  >4.75     43  102
  <=4.75   121  143
> d_s <- tabMarg(lizard, ~diam + species); d_s
     species
diam  anoli dist
  <=4   118  134
  >4     46  111
```

```
> ## Normalize to CPTs if desired (not necessary because
> ## we can always normalize at the end)
> s   <- as.parray(s, normalize="first"); s
species
anoli  dist
0.401 0.599
> h_s <- as.parray(h_s, normalize="first"); h_s
        species
height   anoli  dist
  >4.75  0.262 0.416
  <=4.75 0.738 0.584
> d_s <- as.parray(d_s, normalize="first"); d_s
     species
diam  anoli  dist
  <=4  0.72 0.547
  >4   0.28 0.453
```

14.2 Creating BN from CPTs

```
> cpt.list <- compileCPT(list(s, h_s, d_s)); cpt.list
CPTspec with probabilities:
 P( species )
 P( height | species )
 P( diam | species )
> net <- grain( cpt.list ); net
Independence network: Compiled: FALSE Propagated: FALSE
  Nodes: chr [1:3] "species" "height" "diam"
```

15 Log–linear models

The probability $p_{dhs} = P(D = d, H = h, S = s)$ of a specific configuration is also the probability of a lizard falling into cell (d, h, s) in the contingency table.

We model the *cell probabilities* p_{dhs} by a hierarchical expansion of $\log p_{dhs}$ into interaction terms:

$$\log p_{dhs} = \alpha^0 + \alpha^D_d + \alpha^H_h + \alpha^S_s + \beta^{DH}_{dh} + \beta^{DS}_{ds} + \beta^{HS}_{hs} + \gamma^{DHS}_{dhs}$$

Structure on the model is obtained by setting terms to zero.

saturated model: No terms are set to zero

$$\log p_{dhs} = \alpha^0 + \alpha^D_d + \alpha^H_h + \alpha^S_s + \beta^{DH}_{dh} + \beta^{DS}_{ds} + \beta^{HS}_{hs} + \gamma^{DHS}_{dhs}$$

independence model: All interaction terms are set to zero

$$\log p_{dhs} = \alpha^0 + \alpha^D_d + \alpha^H_h + \alpha^S_s$$

If an interaction term is set to zero then all higher order terms containing that interaction terms must also be set to zero.

For example, if we set $\beta_{dh}^{DH} = 0$ then we must also set $\gamma_{dhs}^{DHS} = 0$.

$$\log p_{dhs} = \alpha^0 + \alpha_d^D + \alpha_h^H + \alpha_s^S + \beta_{ds}^{DS} + \beta_{hs}^{HS}$$

The non–zero interaction terms are the *generators* of the model.

$$\{D, H, S, DS, HS\}$$

Generators contained in higher order generators can be omitted so the generators become

$$\{DS, HS\}$$

corresponding to

$$\log p_{dhs} = \alpha_{ds}^{DS} + \alpha_{hs}^{HS}$$

Because of this log–linear expansions, the models are called *log–linear models*.

15.1 Log–linear models and conditional independence

Instead of taking logs we may write p_{hds} in product form

$$p_{dhs} = q^{DS}(d, s)q^{HS}(h, s)$$

and this is in some connections useful.

For example, the *factorization criterion* gives directly that $D \perp\!\!\!\perp H \mid S$.

15.2 How to - in R

```
> library(MASS) ## to get loglm
> loglm( ~ diam:height:species, data=lizard)
Call:
loglm(formula = ~diam:height:species, data = lizard)

Statistics:
                  X^2 df P(> X^2)
Likelihood Ratio   0  0        1
Pearson            0  0        1
> loglm( ~ diam + height + species, data=lizard)
Call:
loglm(formula = ~diam + height + species, data = lizard)

Statistics:
                   X^2 df P(> X^2)
Likelihood Ratio 25.0  4 4.95e-05
Pearson          23.9  4 8.34e-05
```

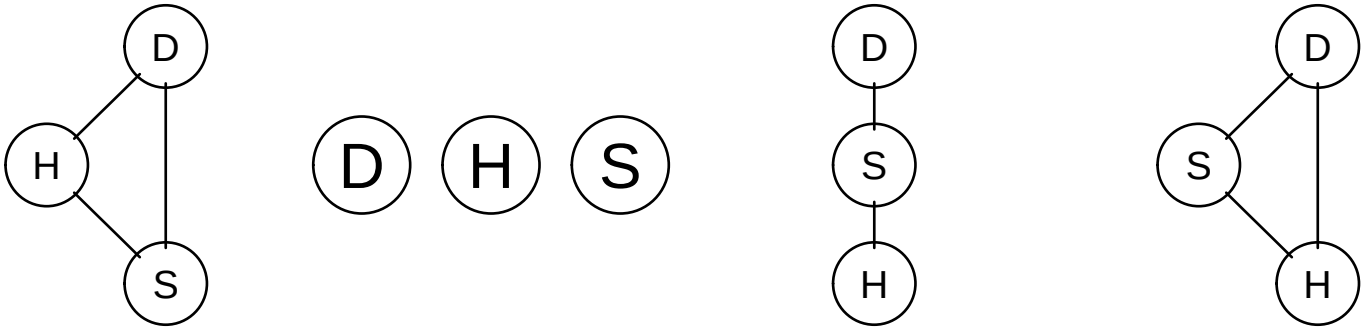
```
> loglm( ~ diam:species + height:species, data=lizard)
Call:
loglm(formula = ~diam:species + height:species, data = lizard)

Statistics:
                     X^2 df P(> X^2)
Likelihood Ratio 2.03  2    0.363
Pearson          2.02  2    0.365
> loglm( ~ diam:species + height:species + diam:height, data=lizard)
Call:
loglm(formula = ~diam:species + height:species + diam:height,
    data = lizard)

Statistics:
                     X^2 df P(> X^2)
Likelihood Ratio 0.149  1    0.699
Pearson          0.151  1    0.698
```

15.3 Dependence graphs

Dependence graph: Variables are nodes; whenever two variables are in the same generator there must be an edge between them:

The `ug()` function

```
> par(mfrow=c(1,4))
> at <- list(node=list(fontsize=5))
> plot( (g1 <- ug(~D:H:S) ), attrs=at)
> plot( (g2 <- ug(~D + H + S) ), attrs=at)
> plot( (g3 <- ug(~D:S + H:S) ), attrs=at)
> plot( (g4 <- ug(~D:S + H:S + D:H) ), attrs=at)
```



The cliques of the dependence graph are sometimes the same as the generators.

```
> str( getCliques( g1 ) )
List of 1
 $ : chr [1:3] "D" "H" "S"
> str( getCliques( g2 ) )
List of 3
 $ : chr "D"
 $ : chr "H"
 $ : chr "S"
> str( getCliques( g3 ) )
List of 2
 $ : chr [1:2] "S" "D"
 $ : chr [1:2] "S" "H"
> str( getCliques( g4 ) ) ## No - not here!
List of 1
 $ : chr [1:3] "D" "S" "H"
```

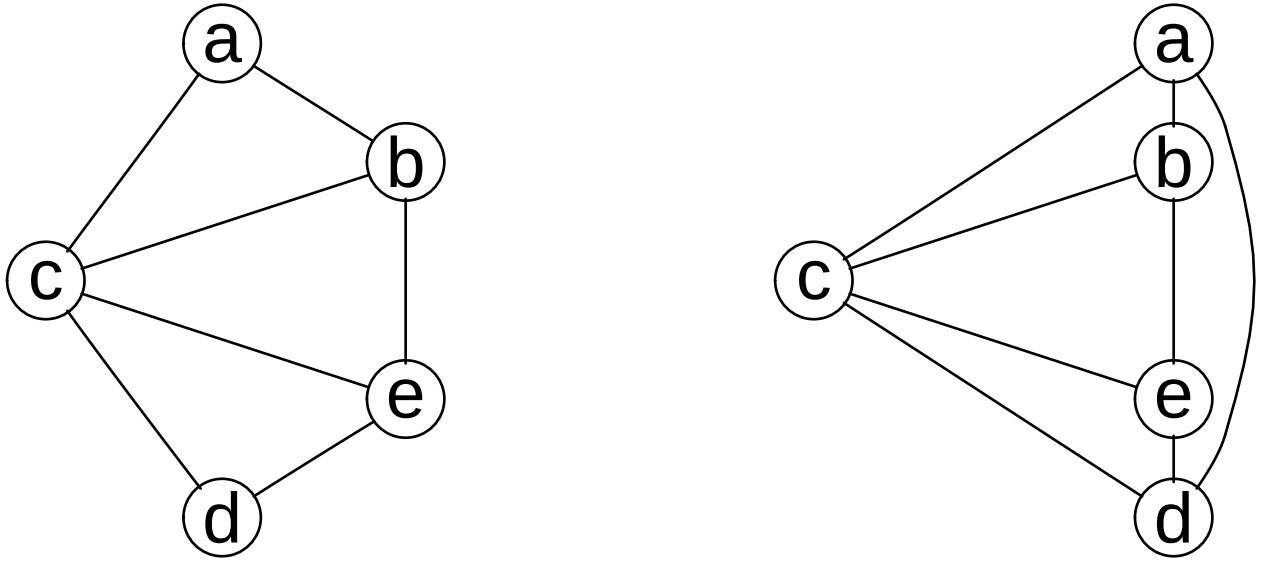
When the generators and the cliques of the dependence graph are the same, the model is said to be a *graphical* log–linear model.

15.4 Decomposable log–linear models

Decomposable log–linear models: graphical models whose dependence graph is *triangulated*

If there are no 4–cycles the graph is triangulated.

```
> par( mfrow=c(1,2))
> plot( ug(~a:b:c+b:c:e+c:d:e) ) # triangulated
> plot( ug(~a:b:c+b:c:e+c:d:e+a:c:d) ) # not triangulated
```



16 Testing for conditional independence

Tests of general conditional independence $u \perp\!\!\!\perp v \mid W$ can be performed with `ciTest()` (a wrapper for calling `ciTest_table()`).

The general syntax of the `set` argument is of the form (u, v, W) where u and v are variables and W is a set of variables.

```
> ciTest(lizard, set=c("diam","height","species"))
Testing diam _|_ height | species
Statistic (DEV):    2.026 df: 2 p-value: 0.3632 method: CHISQ
```

Alternative forms are available:

```
> ciTest(lizard, set=~diam+height+species)
> ciTest(lizard, ~di+he+s)
> ciTest(lizard, c("di","he","sp"))
> ciTest(lizard, c(2,3,1))
```

16.1 What is a CI-test – stratification

Conditional independence $u \perp\!\!\!\perp v \,|\, W$ means independence of u and v for each configuration w^* of W.

Conceptually form a factor S by crossing the factors in W. The test is then a test of the conditional independence $u \perp\!\!\!\perp v \,|\, S$ in a three way table.

The deviance decomposes into independent contributions from each stratum:

$$\begin{aligned} D &= 2\sum_{ijs} n_{ijs} \log \frac{n_{ijs}}{\hat{m}_{ijs}} \\ &= \sum_{s} 2\sum_{ij} n_{ijs} \log \frac{n_{ijs}}{\hat{m}_{ijs}} = \sum_{s} D_s \end{aligned}$$

where the contribution D_s from the sth slice is the deviance for the independence model of u and v in that slice.

The test by `ciTest()` corresponds to the test for removing the edge $\{u, v\}$ from the saturated model with variables $\{u, v\} \cup W$.

```
> lizard
, , species = anoli

     height
diam  >4.75 <=4.75
  <=4    32     86
  >4     11     35

, , species = dist

     height
diam  >4.75 <=4.75
  <=4    61     73
  >4     41     70
```

```
> cit <- ciTest(lizard, set=~diam+height+species, slice.info=T)
> cit
Testing diam _|_ height | species
Statistic (DEV):    2.026 df: 2 p-value: 0.3632 method: CHISQ
> names(cit)
[1] "statistic" "p.value"   "df"          "statname"  "method"
[6] "adjust.df" "varNames"  "slice"
> cit$slice
  statistic p.value df species
1     0.178   0.673  1   anoli
2     1.848   0.174  1    dist
```

The sth slice is a $|u| \times |v|$–table $\{n_{ijs}\}_{i=1\ldots|u|,j=1\ldots|v|}$. The degrees of freedom corresponding to the test for independence in this slice is

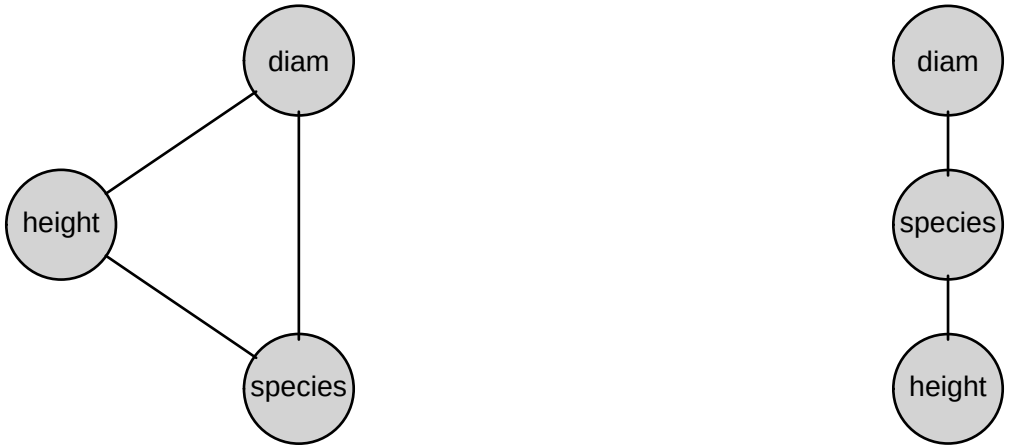
$$df_s = (\#\{i : n_{i\cdot s} > 0\} - 1)(\#\{j : n_{\cdot js} > 0\} - 1)$$

where $n_{i\cdot s}$ and $n_{\cdot js}$ are the marginal totals.

17 Log–linear models with **gRim**

```
> ## Saturated model
> liz1 <- dmod(~height:species:diam, data=lizard)
> ## Shorter form
> liz1 <- dmod(~.^., data=lizard)
> ## backward search among decomposable log-linear models;
> ## focus on deleting edges
> liz2 <- stepwise( liz1, details=1 )
STEPWISE:
 criterion: aic ( k = 2 )
 direction: backward
 type     : decomposable
 search   : all
 steps    : 1000
. BACKWARD: type=decomposable search=all, criterion=aic(2.00), alpha=0
. Initial model: is graphical=TRUE is decomposable=TRUE
  change.AIC   -1.9744 Edge deleted: diam height
```

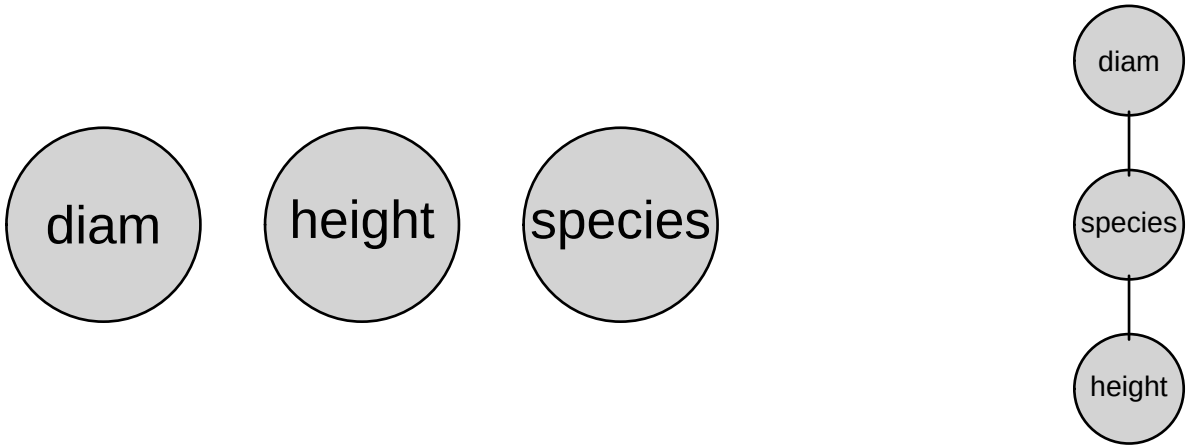
```
> par(mfrow = c(1,2)); plot(liz1); plot( liz2 )
```




```
> liz2
Model: A dModel with 3 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :        1604.43 mdim :    5 aic :      1614.43
 ideviance :          23.01 idf  :    2 bic :      1634.49
 deviance  :           2.03 df   :    2
> formula( liz2 )
~diam * species + height * species
> str( terms( liz2 ) )
List of 2
 $ : chr [1:2] "diam" "species"
 $ : chr [1:2] "height" "species"
```

```
> ## Independence model
> liz0 <- dmod(~height + species + diam, data=lizard)
> ## Shorter form
> liz0 <- dmod(~.^1, data=lizard)
> ## forward search among decomposable log-linear models;
> ## focus on adding edges
> liz3 <- stepwise( liz0, direction = "forward", details=1 )
STEPWISE:
 criterion: aic ( k = 2 )
 direction: forward
 type     : decomposable
 search   : all
 steps    : 1000
. FORWARD: type=decomposable search=all, criterion=aic(2.00), alpha=0.
. Initial model: is graphical=TRUE is decomposable=TRUE
  change.AIC  -10.6062 Edge added: diam species
  change.AIC   -8.4049 Edge added: height species
```

```
> par(mfrow = c(1,2)); plot(liz0); plot( liz3 )
```




```
> liz3
Model: A dModel with 3 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :        1604.43 mdim :    5 aic :      1614.43
 ideviance :          23.01 idf  :    2 bic :      1634.49
 deviance  :           2.03 df   :    2
> formula( liz3 )
~diam * species + height * species
> str( terms( liz3 ) )
List of 2
 $ : chr [1:2] "diam" "species"
 $ : chr [1:2] "height" "species"
```

Some additional models:

```
> ## Saturated model:
> dmod(~.^., data=lizard)
> ## Independence model:
> dmod(~.^1, data=lizard)
> ## Decomposable graphical models {DS, HS}:
> dmod(~height:species + diam:species, data=lizard)
> ## Non-graphical model {DS, HS, DH}:
> dmod(~height:species + diam:species + height:diam, data=lizard)
```

18 The reinis data

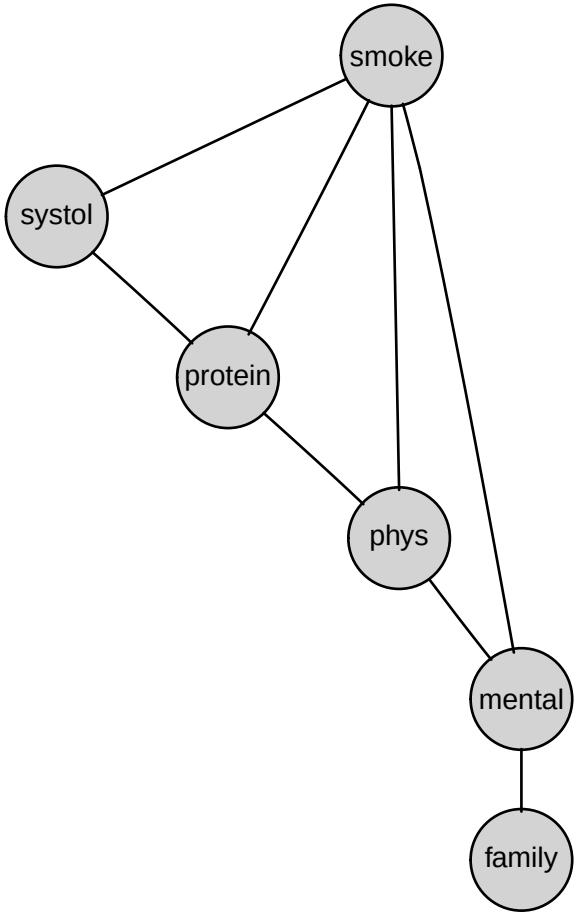
```
> data(reinis, package="gRbase")
> ftable(reinis, row.vars=1:3)
                  systol    y          n
                  protein   y     n    y     n
                  family    y  n  y  n  y  n  y  n
smoke mental phys
y     y      y            44  5 23  7 35  4 24  4
             n           129  9 50  9 109 14 51  5
      n      y           112 21 70 14 80 11 73 13
             n            12  1  7  2  7  5  7  4
n     y      y            40  7 32  3 12  3 25  0
             n           145 17 80 16 67 17 63 14
      n      y            67  9 66 14 33  8 57 11
             n            23  4 13  3  9  2 16  4
```

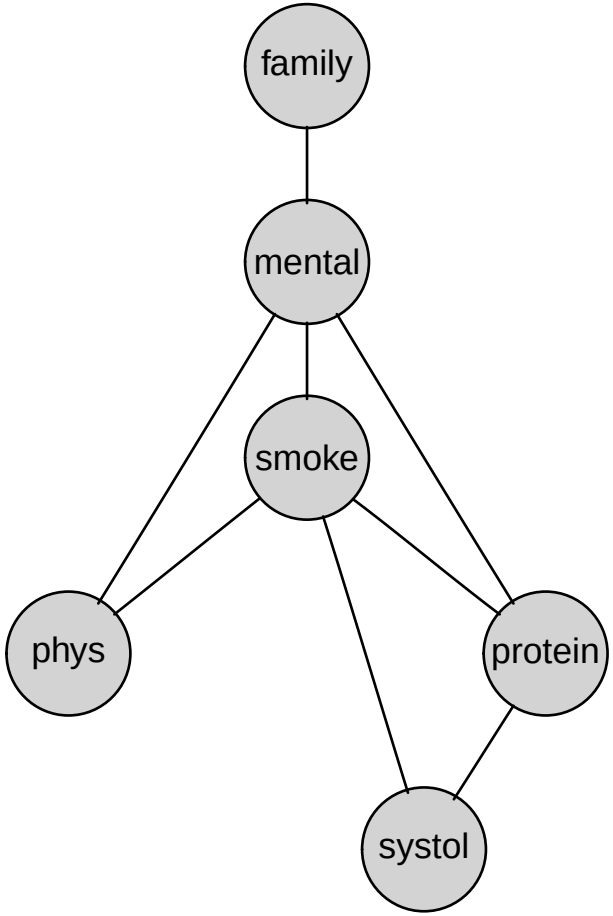
The `reinis` data: A 2^6 contingency table with risk factors of coronary haeart disease. Propspective study of 1841 car–workers in Czechoslovakia, reported in 1981.

```
> rei.s <- dmod(~.^., data=reinis)
> rei.s.2 <- stepwise(rei.s, details=1)
STEPWISE:
 criterion: aic ( k = 2 )
 direction: backward
 type     : decomposable
 search   : all
 steps    : 1000
. BACKWARD: type=decomposable search=all, criterion=aic(2.00), alpha=0
. Initial model: is graphical=TRUE is decomposable=TRUE
  change.AIC  -19.7744 Edge deleted: mental systol
  change.AIC   -8.8511 Edge deleted: phys systol
  change.AIC   -4.6363 Edge deleted: mental protein
  change.AIC   -1.6324 Edge deleted: systol family
  change.AIC   -3.4233 Edge deleted: family protein
  change.AIC   -0.9819 Edge deleted: phys family
  change.AIC   -1.3419 Edge deleted: smoke family
```

```
> rei.1 <- dmod(~.^1, data=reinis)
> rei.1.2 <- stepwise(rei.1, direction = "forward", details=1)
STEPWISE:
 criterion: aic ( k = 2 )
 direction: forward
 type     : decomposable
 search   : all
 steps    : 1000
. FORWARD: type=decomposable search=all, criterion=aic(2.00), alpha=0.
. Initial model: is graphical=TRUE is decomposable=TRUE
  change.AIC -683.9717 Edge added: mental phys
  change.AIC  -25.4810 Edge added: smoke phys
  change.AIC  -15.9293 Edge added: protein mental
  change.AIC  -10.8092 Edge added: systol protein
  change.AIC   -2.7316 Edge added: family mental
  change.AIC   -1.9876 Edge added: mental smoke
  change.AIC  -16.4004 Edge added: protein smoke
  change.AIC  -12.5417 Edge added: systol smoke
```

```
> par(mfrow=c(1,2)); plot(rei.s.2); plot(rei.1.2)
```





19 From graphical model to BN

```
> bn.s.2 <- grain(rei.s.2)
> bn.s.2
Independence network: Compiled: FALSE Propagated: FALSE
  Nodes: chr [1:6] "smoke" "protein" "systol" "phys" "mental" ...
```

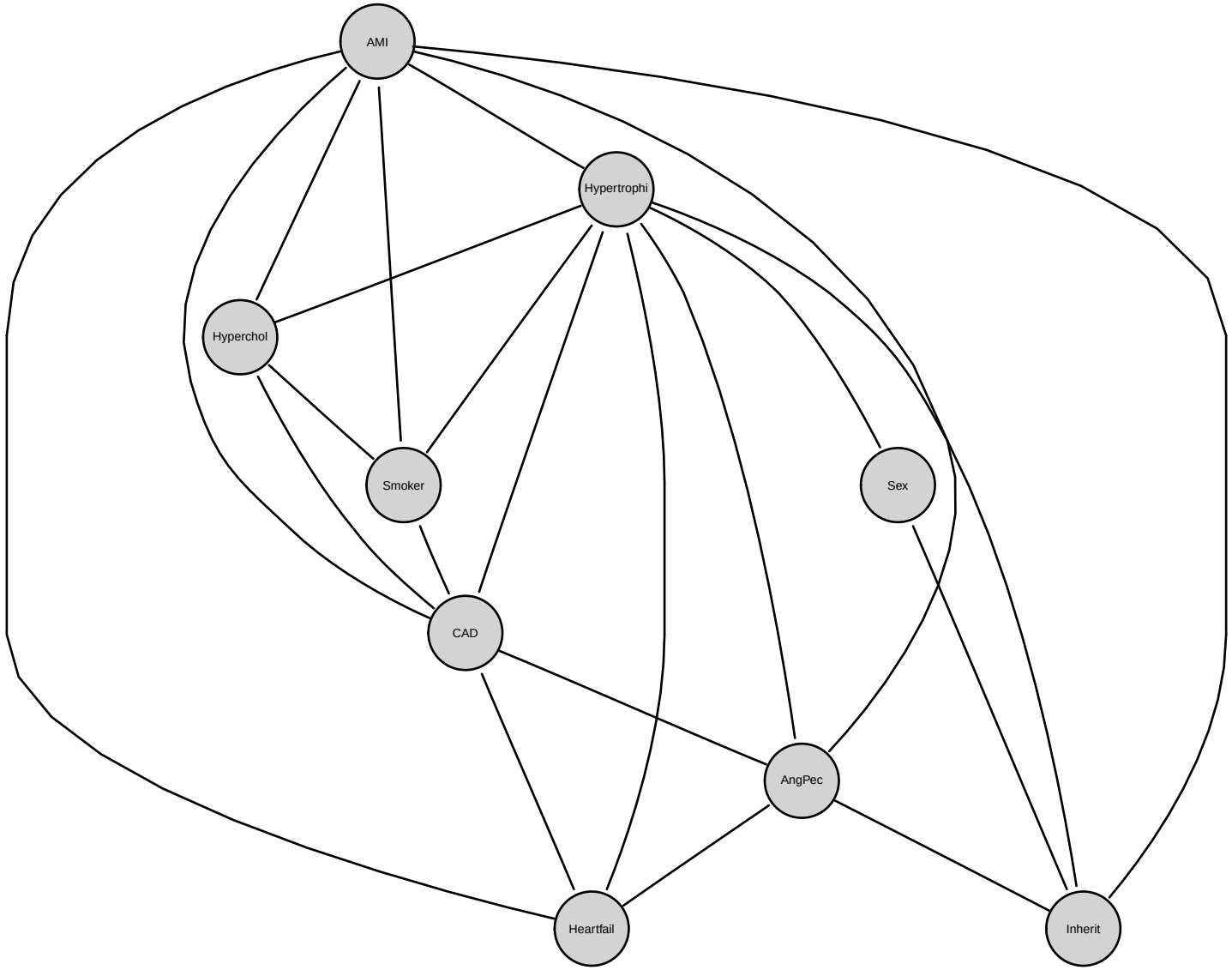
20 The coronary artery disease data

CAD is the disease; the other variables are risk factors and disease manifestations/symptoms.

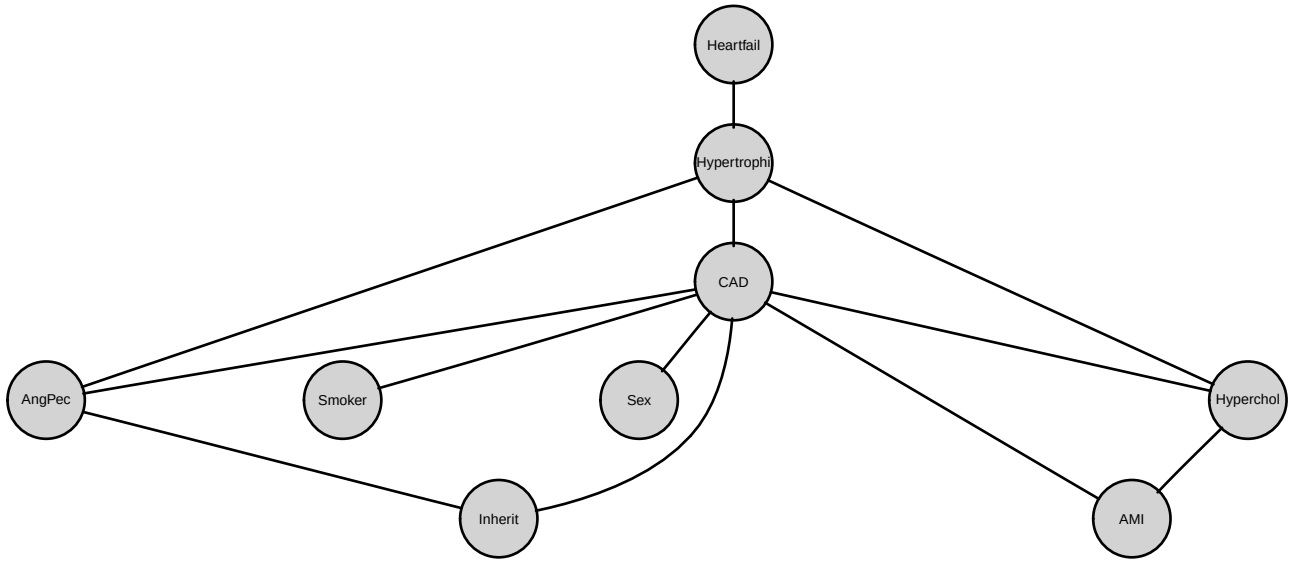
```
> data(cad1, package="gRbase")
> use  <- c(1,2,3,9:14)
> cad1 <- cad1[,use]
> head( cad1, 4 )
     Sex   AngPec         AMI Hypertrophi Hyperchol Smoker Inherit
1   Male     None NotCertain           No         No     No       No
2   Male Atypical NotCertain           No         No     No       No
3 Female     None   Definite           No         No     No       No
4   Male     None NotCertain           No         No     No       No
  Heartfail CAD
1        No  No
2        No  No
3        No  No
4        No  No
```

```
> m.sat <- dmod( ~.^., data=cad1 ) # saturated model
> m.new1 <- stepwise( m.sat, details=0, k=2 ) # use aic
> m.new1
Model: A dModel with 9 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :        2275.24 mdim :   87 aic :      2449.24
 ideviance :         425.03 idf  :   77 bic :      2750.59
 deviance  :         227.02 df   :  680
Notice: Table is sparse
  Asymptotic chi2 distribution may be questionable.
  Degrees of freedom can not be trusted.
  Model dimension adjusted for sparsity : 60
```

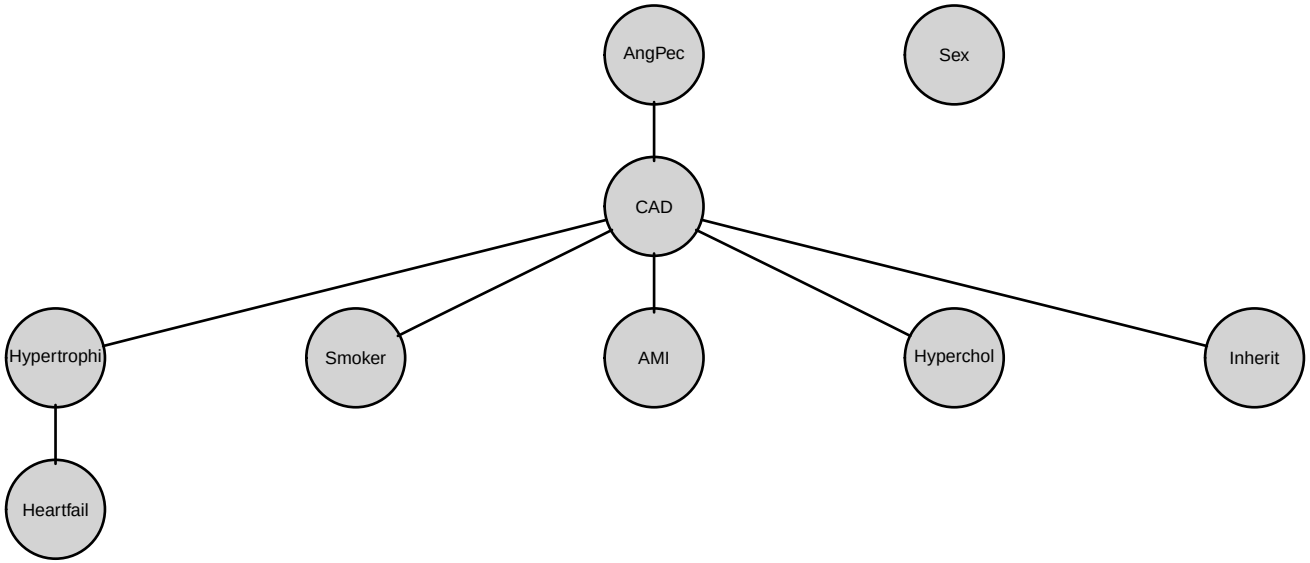
```
> plot( m.new1 )
```


AMI
Hypertrophi
Hyperchol
Smoker
Sex
CAD
AngPec
Heartfail
Inherit

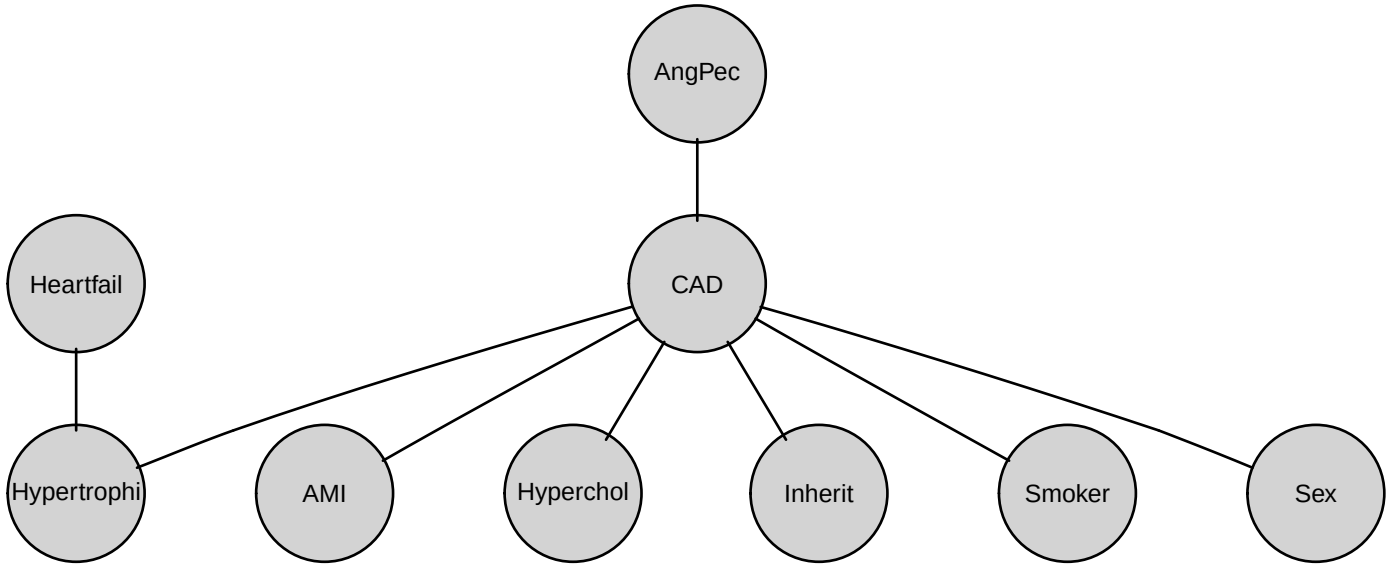
```
> m.ind <- dmod( ~.^1, data=cad1 ) # independence model
> m.new2 <- stepwise( m.ind, direction="forward", details=0, k=2 )
> m.new2
Model: A dModel with 9 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :        2379.06 mdim :   31 aic :      2441.06
 ideviance :         321.21 idf  :   21 bic :      2548.44
 deviance  :         330.84 df   :  736
> plot( m.new2 )
```



```
> m.new3 <- stepwise( m.sat,  k=log(nrow(cad1)) ) # use bic
> m.new3
Model: A dModel with 9 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :        2420.32 mdim :   18 aic :      2456.32
 ideviance :         279.94 idf  :    8 bic :      2518.67
 deviance  :         372.11 df   :  749
> plot( m.new3 )
```



```
> m.new4 <- stepwise( m.ind, direction="forward", k=log(nrow(cad1)) )
> plot( m.new4 )
```



21 From graph and data to network

Create Bayesian networks from model2 (i.e. from (graph,data)):

`smooth`: Add a small number to each cell to avoid configurations with zero probability.

```
> bn1 <- grain( m.new1, smooth=0.01 )
> bn2 <- grain( m.new2, smooth=0.01 )
> bn3 <- grain( m.new3, smooth=0.01 )
> bn4 <- grain( m.new4, smooth=0.01 )
> querygrain( bn1, "CAD")$CAD
CAD
   No   Yes
0.547 0.453
> querygrain( bn1, "CAD",
              evidence=list(AngPec="Typical", Hypertrophi="Yes"))$CAD
CAD
 No Yes
0.6 0.4
```

22 Prediction

Dataset with missing values

```
> data(cad2, package="gRbase")
> use  <- c(1,2,3,9:14)
> cad2 <- cad2[,use]
> head( cad2, 4 )
     Sex   AngPec        AMI Hypertrophi Hyperchol Smoker Inherit
1   Male     None NotCertain           No         No   <NA>      No
2 Female     None NotCertain           No         No   <NA>      No
3 Female     None NotCertain           No        Yes   <NA>      No
4   Male Atypical   Definite           No        Yes   <NA>      No
  Heartfail CAD
1        No  No
2        No  No
3        No  No
4        No  No
```

```
> p1 <- predict(bn1, newdata=cad2, response="CAD")
> head( p1$pred$CAD )
[1] "No"  "No"  "No"  "No"  "No"  "Yes"
> z <- data.frame(CAD.obs=cad2$CAD, CAD.pred=p1$pred$CAD)
> head( z ) # class assigned by highest probability
  CAD.obs CAD.pred
1      No       No
2      No       No
3      No       No
4      No       No
5      No       No
6      No      Yes
> xtabs(~., data=z)
        CAD.pred
CAD.obs No Yes
    No  32   9
    Yes 10  16
```

Let us do so for all models

```
> p <-
      lapply(list(bn1, bn2, bn3, bn4),
         function(bn) predict(bn, newdata=cad2, response="CAD"))
> l <- lapply(p, function(x) x$pred$CAD)
> cls <- as.data.frame(l)
> names(cls) <- c("CAD.pred1","CAD.pred2","CAD.pred3","CAD.pred4")
> cls$CAD.obs <- cad2$CAD
> head(cls)
  CAD.pred1 CAD.pred2 CAD.pred3 CAD.pred4 CAD.obs
1        No        No        No        No      No
2        No        No        No        No      No
3        No        No        No        No      No
4        No        No       Yes       Yes      No
5        No        No        No        No      No
6       Yes        No        No        No      No
```

```
> xtabs( ~ CAD.obs+CAD.pred1, data=cls)
       CAD.pred1
CAD.obs No Yes
    No  32   9
    Yes 10  16
> xtabs( ~ CAD.obs+CAD.pred2, data=cls)
       CAD.pred2
CAD.obs No Yes
    No  35   6
    Yes 10  16
> xtabs( ~ CAD.obs+CAD.pred3, data=cls)
       CAD.pred3
CAD.obs No Yes
    No  32   9
    Yes 10  16
> xtabs( ~ CAD.obs+CAD.pred4, data=cls)
       CAD.pred4
CAD.obs No Yes
    No  33   8
    Yes 10  16
```

23 Winding up

Brief summary:

- We have gone through aspects of the **gRain** package and seen some of the mechanics of probability propagation.
- Propagation is based on factorization of a pmf according to a decomposable graph.
- We have gone through aspects of the **gRim** package and seen how to search for decomposable graphical models.
- We have seen how to create a Bayesian network from the dependency graph of a decomposable graphical model.
- The model search facilities in **gRim** do not scale to large problems; instead it is more useful to consider other approaches for structural learning, see e.g. the **bnlearn** package