Graphical Models and Bayesian Networks *

Søren Højsgaard
Aarhus University, Denmark

August 14, 2011

Contents

*Pre–conference tutorial at the useR! 2011 conference August 16-18 2011 University of Warwick, Coventry, UK

1 Outline of tutorial

Goal: Establish a Bayesian network (BN) for diagnosing coronary artery disease (CAD) from a contingency table.

- Bayesian networks
- Introduce the `gRain` package
- Conditional independence restrictions and dependency graphs
- Probability propagation
- Log–linear, graphical and decomposable models for contingency tables
- Introduce the `gRim` package
- Use `gRim` for model selection
- Convert decomposable model to Bayesian network.

---

2 Book: Graphical Models with R

The book

> "Graphical Models with R"

(in Springer's useR series) by Højsgaard, Edwards and Lauritzen will be available in the Spring, 2012.

---

3 R–packages

- We shall in this tutorial use the R–packages `gRbase`, `gRain` and `gRim`.
- `gRbase` and `gRain` have been on CRAN for some years now and are fairly stable.
- `gRim` is a recent package and is likely to undergo larger changes.
- If you discover bugs etc. in any of these 3 packages, please send me an e–mail; preferably with a small reproducible example.
- We shall be using the `Rgraphviz` package for plotting graphs. This requires that the standalone program `Graphviz` is installed on your computer.

---

4 The coronary artery disease data

Goal: Build BN for diagnosing coronary artery disease (CAD) from these data:

```
data(cad1)
head(cad1)
```

```
     Sex   AngPec        AMI QWave QWavecode    STcode STchange SuffHeartF
1   Male     None NotCertain    No    Usable    Usable       No         No
2   Male Atypical NotCertain    No    Usable    Usable       No         No
3 Female     None   Definite    No    Usable    Usable       No         No
4   Male     None NotCertain    No    Usable Nonusable       No         No
5   Male     None NotCertain    No    Usable Nonusable       No         No
6   Male     None NotCertain    No    Usable Nonusable       No         No
  Hypertrophi Hyperchol Smoker Inherit Heartfail CAD
1          No        No     No       No        No  No
2          No        No     No       No        No  No
3          No        No     No       No        No  No
4          No        No     No       No        No  No
5          No        No     No       No        No  No
6          No        No     No       No        No  No
```

Validate model by prediction of CAD using these data. Notice: incomplete information.

```
data(cad2)
head(cad2)
```

```
     Sex   AngPec        AMI QWave QWavecode    STcode STchange SuffHeartF
1   Male     None NotCertain    No    Usable    Usable      Yes        Yes
2 Female     None NotCertain    No    Usable    Usable      Yes        Yes
3 Female     None NotCertain    No Nonusable Nonusable       No         No
4   Male Atypical   Definite    No    Usable    Usable       No        Yes
5   Male     None NotCertain    No    Usable    Usable      Yes         No
6   Male     None   Definite    No    Usable Nonusable       No         No
  Hypertrophi Hyperchol Smoker Inherit Heartfail CAD
1          No        No   <NA>       No        No  No
2          No        No   <NA>       No        No  No
3          No       Yes   <NA>       No        No  No
4          No       Yes   <NA>       No        No  No
5         Yes       Yes   <NA>       No        No  No
6          No        No     No    <NA>        No  No
```

---

5 A small worked example BN

Consider the following narrative:

Having flu (F) may cause an elevated body temperature (T) (fever). An elevated body temperature may cause a headache (H).

Illustrate this narrative by DIRECTED ACYCLIC GRAPH (or DAG):

```
plot(dag(~F+T:F+H:T))
```

We have a UNIVERSE consisting of the variables $V = \{F, T, H\}$ which all have a finite STATE SPACE . (Here all are binary).

Corresponding to V there is a RANDOM VECTOR $X = X_V = (X_F, X_T, X_H)$ where $x = x_V = (x_F, x_T, x_H)$ denotes a specific CONFIGURATION .

For $A \subset V$ we have the random vector $X_A = (X_v; v \in A)$ where a configuration is denoted x_A.

We define a joint pmf for X as

$$p_X(x) = p_{X_F}(x_F)p_{X_T|X_F}(x_T|x_F)p_{X_H|X_T}(x_H|x_T) \tag{1}$$

We shall allow a simpler notation. Let A and B be disjoint subsets of V. We may then use one of the forms:

$$p_{X_A|X_B}(x_A|x_B) = p_{A|B}(x_A|x_B) = p(x_A|x_B) = p(A|B) = (A|B)$$

Notice: In the special case where $A = V$ we then allow to write $p(V)$ rather than $p_V(x_V)$

Hence (1) may be written

$$p(V) = p(F)p(T|F)p(H|T)$$

Notice: By definition of CONDITIONAL PROBABILITY we have from BAYES FORMULA that

$$p(V) = p(F)p(T|F)p(H|T, F)$$

So the fact that in

$$p(V) = p(F)p(T|F)p(H|T)$$

we have $p(H|T)$ rather than $p(H|T, F)$ reflects the MODEL ASSUMPTION that if temperature (fever status) is known then knowledge about flu will provide no additional information about headache.

We say that headache is CONDITIONALLY INDEPENDENT of flu given temperature.

Given a FINDING or EVIDENCE that a person has headache we may now calculate e.g. the probability of having flu, i.e. $p(F = yes|H = yes)$.

In this small example we can compute everything in a brute force way using table operation functions from `gRbase`.

5.1 Specification of conditional probability tables

We may specify $p(F)$, $p(T|F)$ and $p(H|T)$ as tables

```
p.F <- parray("T", levels=2, values=c(.01,.99))
```

```
T
  T1   T2
0.01 0.99
```

```
p.TgF <- parray(c("T","F"), levels=c(2,2), values=c(.95,.05, .01,.99))
```

```
    F
T      F1   F2
  T1 0.95 0.01
  T2 0.05 0.99
```

```
p.HgT <- parray(c("H","T"), levels=c(2,2), values=c(.8,.2, .1,.9))
```

```
    T
H     T1  T2
  H1 0.8 0.1
  H2 0.2 0.9
```

5.2 Brute force computations

A brute force approach is as follows: 1) First calculate joint distribution:

```
p.V <- tableMult(tableMult(p.F, p.TgF), p.HgT)
as.data.frame.table(p.V)
```

```
   H  T  F    Freq
1 H1 T1 F1 0.00760
2 H2 T1 F1 0.00190
3 H1 T2 F1 0.00495
4 H2 T2 F1 0.04455
5 H1 T1 F2 0.00008
6 H2 T1 F2 0.00002
7 H1 T2 F2 0.09801
8 H2 T2 F2 0.88209
```

2) Then calculate the marginal distribution

```
p.FT <- tableMargin(p.V, margin=c('F','T'))
as.data.frame.table(p.FT)
```

```
   F  T   Freq
1 F1 T1 0.0095
2 F2 T1 0.0001
3 F1 T2 0.0495
4 F2 T2 0.9801
```

3) Then calculate conditional distribution

```
p.T <- tableMargin(p.FT, margin='T')
p.FgT <- tableDiv(p.FT, p.T)
p.FgT
```

```
    F
T            F1         F2
  T1 0.98958333 0.01041667
  T2 0.04807692 0.95192308
```

So $p(F = yes|H = yes) = 0.989$.

However, this scheme is computationally prohibitive in large networks.

6 Bayesian Network (BN) basics

- A BAYESIAN NETWORK is a often understood to be graphical model based on a DIRECTED ACYCLIC GRAPH (a DAG).

- A BN typically will typically satisfy CONDITIONAL INDEPENDENCE restrictions which enables computations of updated probabilities for states of unobserved variables to be made very EFFICIENTLY .
- The DAG only is used to give a simple and transparent way of specifying a probability model.
- The computations are based on exploiting conditional independencies in an undirected graph.
- Therefore, methods for building UNDIRECTED GRAPHICAL MODELS can just as easily be used for building BNs.

6.1 DAGs and probabilities

Given a DIRECTED ACYCLIC GRAPH (a DAG) with vertices V we may define a pmf for X_V as

$$p(x) = \prod_{v \in V} p(x_v | x_{pa(v)})$$

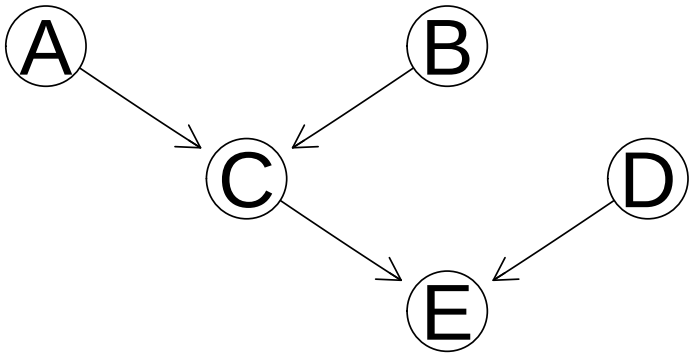
which we write in a simpler notation as

$$p(V) = \prod_{v \in V} p(v|pa(v))$$

Notice: We specify a complex multivariate distribution $p(V)$ by multiplying simple univariate conditionals $p(v|pa(v))$.

Notice: In the discrete case, each conditional distribution is specified as a table called a CONDITIONAL PROBABILITY TABLE or a CPT for short.

```
plot(dag(~A+B+C|A:B+D+E|C:D))
```



$$p(V) = p(A)p(B)p(C|A,B)p(D)p(E|C,D)$$

6.2 The chest clinic narrative

Lauritzen and Spiegehalter (1988) presents 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 neither does the presence or absence of dyspnoea."

The universe is

$$V = \{\texttt{Asia}, \texttt{Tub}, \texttt{Smoke}, \texttt{Lung}, \texttt{Either}, \texttt{Bronc}, \texttt{X-ray}, \texttt{Dysp}\}$$

A formalization of this narrative is as follows:

The DAG in Figure 1 now corresponds to a factorization of the joint probability function as

$$p(V) = p(A)p(T|A)p(S)p(L|S)p(B|S)p(E|T,L)p(D|E,B)p(X|E). \tag{2}$$

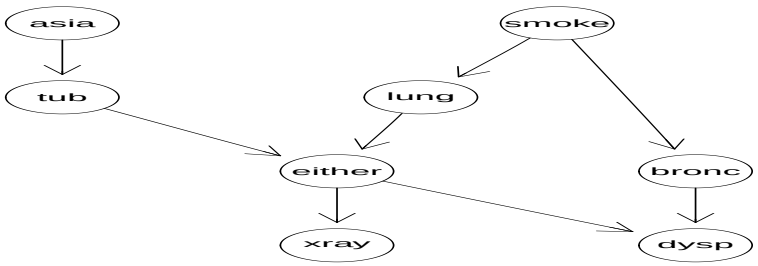


Figure 1: The directed acyclic graph corresponding to the chest clinic example.

6.3 Notation

Let V be a set of variables.

We consider a DISCRETE RANDOM VECTOR $X = (X_v; v \in V)$.

Each component X_v has a finite state space $\mathcal{X}_v$

A CONFIGURATION of X is denoted by x.

The set of configurations is $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ if there are $d = |V|$ variables.

For $A \subset V$ we correspondingly have $X_A = (X_v; v \in A)$.

A CONFIGURATION of X_A is denoted by x_A. The set of configurations of X_A is $\mathcal{X}_A$.

For $A \subset V$ we let $\psi_A(x_A)$ denote a non–negative function which depends on x only through x_A. We call such a function a POTENTIAL . We shall sometimes skip x_A and simply write ψ_A.

6.4 Findings and queries

- Suppose we are given the FINDING that a person has recently visited Asia and suffers from dyspnoea, i.e. $A =$ yes and $D =$ yes. Generally denote findings as $E = e^*$
- Interest may be in the conditional distributions $p(L|e^*)$, $p(T|e^*)$ and $p(B|e^*)$, or possibly in the joint (conditional) distribution $p(L, T, B|e^*)$.
- Interest might also be in calculating the probability of a specific event, e.g. the probability of seeing a specific evidence, i.e. $p(E = e^*)$.
- A brute–force approach is to calculate the joint distribution by carrying out the table multiplications and then marginalizing.
- This is doable in this example (the joint distribution will have $2^8 = 256$ states) but with 100 binary variables the state space will have 2^{100} states. That is prohibitive.
- The `gRain` package implements a computationally much more efficient scheme.

---

7 An introduction to the `gRain` package

Specify chest clinic network.

```
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)
```

```
plist <- compileCPT(list(a, t.a, s, l.s, b.s, e.lt, x.e, d.be))
bnet    <- grain(plist)
bnet
```

```
Independence network: Compiled: FALSE Propagated: FALSE
 Nodes: chr [1:8] "asia" "tub" "smoke" "lung" "bronc" "either" ...
```

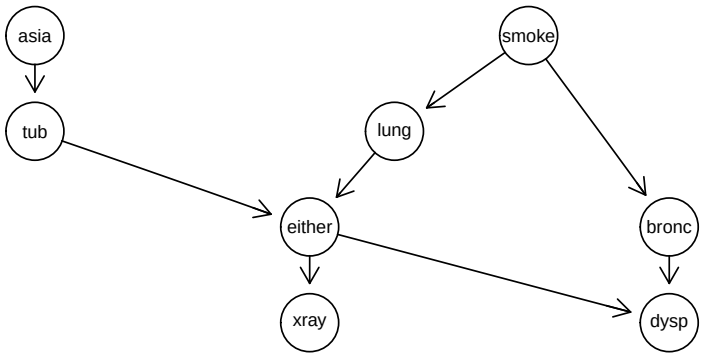
```
plist
```

```
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 )
```

```
plist$tub
```

```
     asia
tub    yes   no
  yes 0.05 0.01
  no  0.95 0.99
```

```
plot(bnet)
```



7.1 Queries

```
querygrain(bnet, nodes=c('lung', 'tub', 'bronc'))
```

```
$tub
tub
   yes     no
0.0104 0.9896

$lung
lung
  yes    no
0.055 0.945

$bronc
bronc
 yes   no
0.45 0.55
```

7.2 Setting findings

```
bnet.f <- setFinding(bnet, nodes=c('asia', 'dysp'), state=c('yes','yes'))
bnet.f
```

```
Independence network: Compiled: TRUE Propagated: TRUE
 Nodes: chr [1:8] "asia" "tub" "smoke" "lung" "bronc" "either" ...
 Findings: chr [1:2] "asia" "dysp"
```

```
pFinding(bnet.f)
```

```
[1] 0.004501375
```

7.3 Queries – II

```
querygrain(bnet.f, nodes=c('lung', 'tub', 'bronc'))
```

```
$tub
tub
       yes         no
```

```
0.08775096 0.91224904

$lung
lung
       yes         no
0.09952515 0.90047485

$bronc
bronc
      yes        no
0.8114021 0.1885979
```

```
querygrain(bnet.f, nodes=c('lung', 'tub', 'bronc'), type='joint')
```

```
, , bronc = yes

     tub
lung          yes         no
  yes 0.003149038 0.05983172
  no  0.041837216 0.70658410

, , bronc = no

     tub
lung          yes         no
  yes 0.001827219 0.03471717
  no  0.040937491 0.11111605
```

7.4 Probability of a finding

```
getFinding(bnet.f)
```

```
Finding:
     variable state
[1,] asia     yes
[2,] dysp     yes
Pr(Finding)= 0.004501375
```

```
pFinding(bnet.f)
```

```
[1] 0.004501375
```

---

8 Conditional independence restrictions

Efficient compuations in BNs are based on exploiting CONDITIONAL INDEPENDENCE restrictions.

- X and Y are conditionally independent given Z (written $X \perp\!\!\!\perp Y|Z$) if

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

  – or equivalently

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

- So if $Z = z$ is known then knowledge of X will provide no additional knowledge of Y.
- A general condition is the FACTORIZATION CRITERION : $X \perp\!\!\!\perp Y|Z$ if

$$p(x, y, z) = g(x, z)h(y, z)$$

  for non–negative functions $g()$ and $h()$.

8.1 Dependence graph

Given variables V, let $\mathcal{A} = \{a_1, \ldots, a_Q\}$ be a collection of subset of V.

Suppose

$$p(x) = \prod_{a \in \mathcal{A}} \phi_a(x_a)$$

where $\phi_a()$ is a non–negative function of x_a.

The DEPENDENCE GRAPH for p has vertices V and undirected edges given as follows:

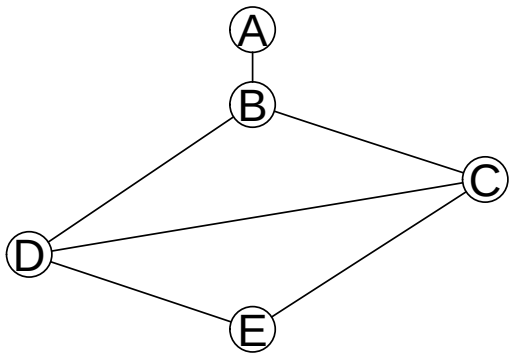
There is an edge between α and β if $\{\alpha, \beta\}$ is in one of the sets $a \in \mathcal{A}$.

Suppose

$$p(x) = \phi_{AB}(x_{AB})\psi_{BCD}(x_{BCD})\psi_{CDE}(x_{CDE})$$

Then the DEPENDENCE GRAPH for p is given as follows:

```
plot(ug(~A:B+B:C:D+C:D:E))
```



8.2 Reading conditional independencies – global Markov property

Conditional independencies can be read off the dependence graph:

- Recall basic factorization p:

$$p(x) = \phi_{AB}(x_{AB})\psi_{BCD}(x_{BCD})\psi_{CDE}(x_{CDE})$$

- Recall factorization criterion $X \perp\!\!\!\perp Y|Z$ if

$$p(x, y, z) = g(x, z)h(y, z)$$

- Global Markov Property : If X and Y are separated by Z in the dependence graph $\mathcal{G}$ then $X \perp\!\!\!\perp Y|Z$.
- Example: $(D, E) \perp\!\!\!\perp A|(B, C)$:

  Proof:

$$p(x) = \left[\phi_{AB}(x_{AB})\psi_{BCD}(x_{BCD})\right]\psi_{CDE}(x_{CDE}) = g(x_{ABCD})h(x_{CDE})$$

8.3 Dependence graph for chest clinic example

Recall chest clinic model

$$p(V) = p(A)p(T|A)p(S)p(L|S)p(B|S)p(E|T, L)p(D|E, B)p(X|E).$$

Think of conditional probilities as potentials and rewrite as:

$$p(V) = \psi(A)\psi(T, A)\psi(S)\psi(L, S)\psi(B, S)\psi(E, T, L)\psi(D, E, B)\psi(X, E).$$

Absorb lower order terms into higher order terms:

$$p(V) = \psi(T, A)\psi(L, S)\psi(B, S)\psi(E, T, L)\psi(D, E, B)\psi(X, E).$$

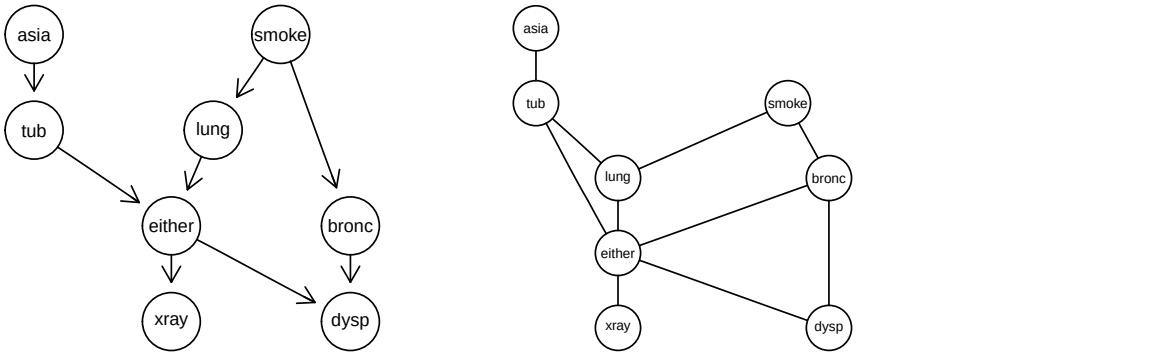
This factorial form implies certain CONDITIONAL INDEPENDENCE restrictions that can be read from the MORAL GRAPH .

Given DAG, the MORAL GRAPH is obtained by 1) marrying parents and 2) dropping directions. Called MORALIZATION .

Recall:

$$p(V) = \psi(T, A)\psi(L, S)\psi(B, S)\psi(E, T, L)\psi(D, E, B)\psi(X, E).$$

```
par(mfrow=c(1,2))
plot(bnet$dag)
plot(moralize(bnet$dag))
```



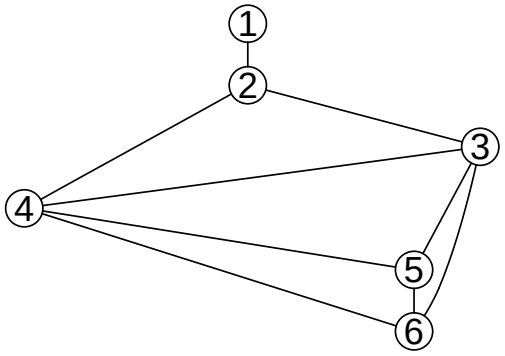
---

9 Decomposable graphs and junction trees

Undirected decomposable graphs play a central role.

A CLIQUE of a graph is a MAXIMAL COMPLETE SUBGRAPH .

```
plot(ug(~1:2+2:3:4+3:4:5:6))
```

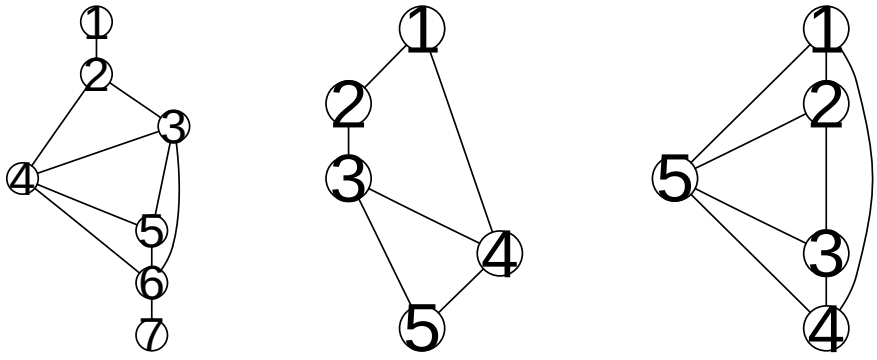


The cliques are $\{1,2\}$, $\{2,3,4\}$, $\{3,4,5,6\}$

9.1 Decomposable graphs

A graph is DECOMPOSABLE (or TRIANGULATED) if it contains no cycles of length ≥ 4.

```
par(mfrow=c(1,3))
plot(ug(~1:2+2:3:4+3:4:5:6+6:7))
plot(ug(~1:2+2:3+3:4:5+4:1))
plot(ug(~1:2:5+2:3:5+3:4:5+4:1:5))
```

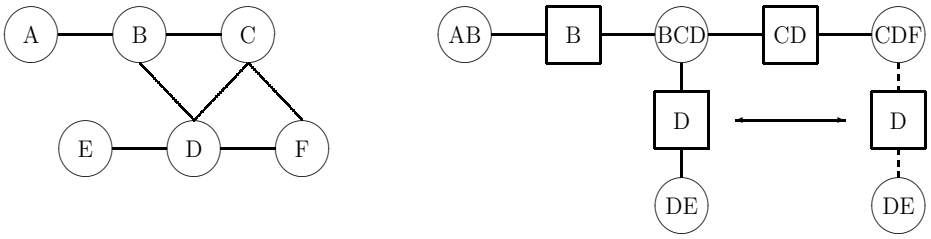


Left: decomposable; center: not decomposable; right: not decomposable

9.2 Junction tree

Result: A graph is DECOMPOSABLE iff it can be represented by a JUNCTION TREE (not unique).

For any two cliques C and D, $C \cap D$ is a subset of every node between them in the junction tree.



9.3 The key to message passing

Suppose

$$p(x) = \prod_{C:cliques} \psi_C(x_C)$$

where C are the cliques of a decomposable graph.

We may write p in a CLIQUE POTENTIAL REPRESENTATION

$$p(x) = \frac{\prod_{C:cliques} \psi_C(x_C)}{\prod_{S:separators} \psi_S(x_S)}$$

The terms are called POTENTIALS ; the representation is not unique.

Potential representation easy to obtain:

- Set all $\psi_C(x_C) = 1$ and all $\psi_S(x_S) = 1$
- Assign each conditional $p(x_v|x_{pa(v)})$ to a potential ψ_C for a clique C containing $v \cup pa(v)$ by

$$\psi_C(x_C) \leftarrow \psi_C(x_C)p(x_v|x_{pa(v)})$$

Using local computations we can manipulate the potentials to obtain CLIQUE MARGINAL REPRESENTATION :

$$p(x) = \frac{\prod_{C:cliques} p_C(x_C)}{\prod_{S:separators} p_S(x_S)}$$

1. First until the potentials contain the clique and separator marginals, i.e. $\psi_C(x_C) = p_C(x_C)$.
2. Next until the potentials contain the clique and separator marginals conditional on a certain set of findings, i.e. $\psi_C(x_C, e^*) = p_C(x_C|e^*)$.

Done by MESSAGE PASSING in JUNCTION TREE .

Notice: We do not want to carry out the multiplication above. Better to think about that we have a representation of p as

$$p \equiv \{p_C, p_S; C : cliques, S : separators\}$$

9.4 Message passing – a simple example

```
plot(dag(~F+T:F+H:T))
```

Define probability distribution according to DAG (where $H \perp\!\!\!\perp F|T$):

$$p = p(F, H, T) = p(F)p(T|F)p(H|T)$$

Want to find $p(F|H = H1)$ (i.e. probability of having flu given headache).

9.5 Obtaining marginal tables

Junction tree:

Setting $\psi_{FT} = p(F)p(T|F)$, $\psi_{TH} = p(H|T)$ and $\psi_T = 1$ gives

$$p(F, H, T) = \frac{\psi_{FT}\psi_{TH}}{\psi_T}$$

Choose any node as ROOT ; we pick TH.

1. Work inwards towards root (i.e. from FT towards TH):
   Set $\psi^*_T = \sum_F \psi_{FT}$. Then

$$p(F,H,T) = \psi_{FT}\frac{1}{\psi^*_T}\left[\frac{\psi^*_T}{\psi_T}\psi_{TH}\right] = \frac{\psi_{FT}\psi^*_{TH}}{\psi^*_T}$$

   Now we have ψ^*_{TH} is the marginal probability $p(T,H)$:

$$\psi^*_{TH} = \sum_F p(F,T,H) = p(T,H) \quad \checkmark$$

2. Work outwards from root (i.e. from TH towards FT):
   Set $\psi^\dagger_T = \sum_H \psi^*_{TH}$. Since $\psi^*_{TH} = p(T,H)$ we have

$$\psi^\dagger_T = p(T) \quad \checkmark$$

   Then:

$$p(F,T,H) = \left[\psi_{FT}\frac{\psi^\dagger_T}{\psi^*_T}\right]\frac{1}{\psi^\dagger_T}\psi^*_{TH} = \psi^*_{FT}\frac{1}{\psi^\dagger_T}\psi^*_{TH}$$

   But now

$$\psi^*_{FT} = p(F,T) \quad \checkmark$$

In other words: We have established

$$p(F,T,H) = \frac{p_{FT}p_{TH}}{p_T}$$

9.6 Propagating findings

Suppose H can take values $H1$ and $H2$ and the finding is $H = H1$ (headache=yes). Want $p(F|H = H1)$.

This finding is propagated as follows:

- Pick any node containg H in the junction tree (e.g. the node TH).
- Set any entry in ψ_{TH} which is inconsistent with $H = H1$ equal to 0. This yields a new potential, say $\tilde{\psi}_{TH}$ and we have

$$p(F,T|H = H1) \propto P(F,T,H = H1) = \frac{\psi_{FT}\tilde{\psi}_{TH}}{\psi_T}$$

- Now repeat the steps above and we get

$$p^*(F,T,H) = \frac{p^*_{FT}p^*_{TH}}{p^*_T}$$

in a probability distribution where any configuration (F,T,H) with $H \neq H1$ has probability 0.

9.7 Message passing in junction tree

- Pick any vertex as root.
- Inwards: Root ask neighbours who ask neighbours who ask neighbours ... for information.
- Outwards: Root sends information to neighbours who send information to neighbours ...
- We are done!

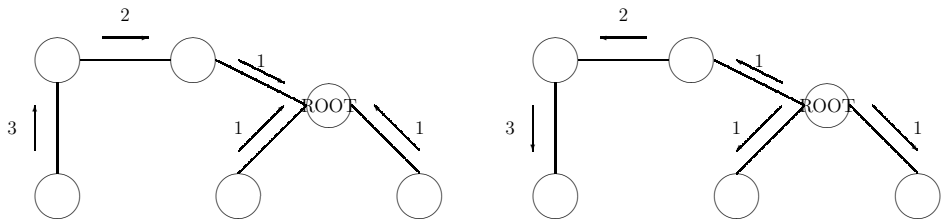


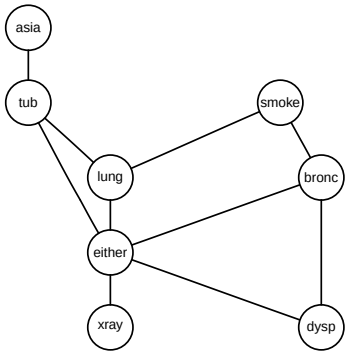
Figure 2: Message passing in junction tree

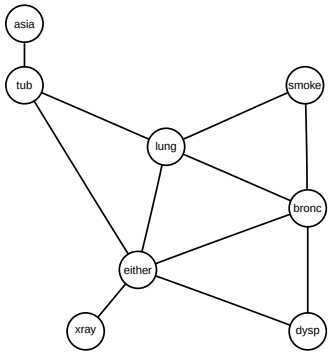
9.8 Triangulation

The dependence graph for the chest clinic example is not decomposable (it contains 4–cycles) so the message passing scheme is not directly applicable.

But, we can add edges, so called FILL–INS to the the dependence graph to make the graph decomposable. This is called TRIANGULATION :

```
par(mfrow=c(1,2))
plot(moralize(bnet$dag))
plot(triangulate(moralize(bnet$dag)))
```





DAG:

$$p(V) = p(A)p(T|A)p(S)p(L|S)p(B|S)p(D|E,B)p(E|T,L)p(X|E).$$

Dependence graph (moral graph):

$$p(V) = \psi(T,A)\psi(L,S)\psi(B,S)\psi(D,E,B)\psi(E,T,L)\psi(X,E).$$

Triangulated graph:

$$p(V) = \psi(T,A)\psi(L,S,B)\psi(L,E,B)\psi(D,E,B)\psi(E,T,L)\psi(X,E)$$

where

$$\psi(L,S,B) = \psi(L,S)\psi(B,S) \quad \phi(L,E,B) \equiv 1$$

Notice: We have not changed the fundamental model by these steps, but some conditional independencies are concealed in the triangulated graph.

But the triangulated graph factorization allows efficient calculations. ✓

9.9 Fundamental operations in `gRain`

Fundamental operations in `gRain` so far:

- Network specification: **grain()** Create a network from list of conditional probability tables; and do a few other things.
- Set findings: **setFinding()** : Set the values of some variables.
- Ask queries: **querygrain()** : Get updated beliefs (conditional probabilities given findings) of some variables

Under the hood there are two additional operations:

- Compilation: **compile()** Create a clique potential representation (and a few other steps)
- Propagation: **propagate()** Turn clique potentials into clique marginals.

These operations must be made before querygrain() can be called but querygrain() will make these operations if necessary.

---

10 Summary of the BN part

We have used a DAG for specifying a complex stochastic model through simple conditional probabilities

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

Afterwards we transfer the model to a factorization over the cliques of a decomposable undirected graph

$$p(V) = \{ \prod_{C:cliques} \psi_C(C)\}/\{ \prod_{S:separators} \psi_S(S)\}$$

It is through the decomposable graph the efficient computation of probabilities takes place.

We then forget about the DAG part and the conditional probability tables.

Therefore, we may skip the DAG part and find the decomposable graph and corresponding clique potentials from data.

---

11 Contingency tables

In a study of lizard behaviour, characteristics of 409 lizards were recorded, namely species (S), perch diameter (D) and perch height (H).

```
data(lizardRAW, package="gRbase")
head(lizardRAW)
```

```
  diam height species
1   >4  >4.75    dist
2   >4  >4.75    dist
3  <=4 <=4.75   anoli
4   >4 <=4.75   anoli
5   >4 <=4.75    dist
6  <=4 <=4.75   anoli
```

```
dim(lizardRAW)
```

```
[1] 409   3
```

We have $V = \{D, H, S\}$.

We may summarize data in a CONTINGENCY TABLE with cells (dhs) and counts n_{dhs} given by:

```
data(lizard, package="gRbase")
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
```

11.1 Notation

Recall the notation:

Let V be a set of variables.

We consider a DISCRETE RANDOM VECTOR $X = (X_v; v \in V)$.

Each component X_v has a finite state space $\mathcal{X}_v$

A CONFIGURATION of X is denoted by x.

The set of configurations is $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ if there are $d = |V|$ variables.

For $A \subset V$ we correspondingly have $X_A = (X_v; v \in A)$.

A CONFIGURATION of X_A is denoted by x_A. The set of configurations of X_A is $\mathcal{X}_A$.

For $A \subset V$ we let $\psi_A(x_A)$ denote a non–negative function which depends on x only through x_A. We call such a function a POTENTIAL . We shall sometimes skip x_A and simply write ψ_A.

A CONFIGURATION x is also a CELL in a CONTINGENCY TABLE . The COUNTS in the cell is denoted $n(x)$ and the total number of observations in denoted n.

For $A \subset V$ we correspondingly have a MARGINAL TABLE with counts $n(x_A)$.

11.2 Log–linear models

We are interested in modelling the CELL PROBABILITIES p_{dhs}.

Commonly done by a hierarchical expansion of log–cell–probabilities 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 interaction terms to zero following the principle that 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 $\gamma_{dhs}^{DHS} = 0$.

The non–zero interaction terms are the generators of the model. Setting $\beta_{dh}^{DH} = \gamma_{dhs}^{DHS} = 0$ the generators are

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

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

$$\{DS, HS\}$$

corresponding to

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

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

$$p_{dhs} = \psi_{ds}^{DS} \psi_{hs}^{HS}$$

The FACTORIZATION CRITERION gives directly that $D \perp\!\!\!\perp H|S$.

More generally the GENERATING CLASS of a log–linear model is a set $\mathcal{A} = \{A_1, \ldots, A_Q\}$ where $A_q \subset V$.

This corresponds to

$$p(x) = \prod_{A \in \mathcal{A}} \phi_A(x_A)$$

where ϕ_A is a potential, a function that depends on x only through x_A.

Under MULTINOMIAL SAMPLING the likelihood is

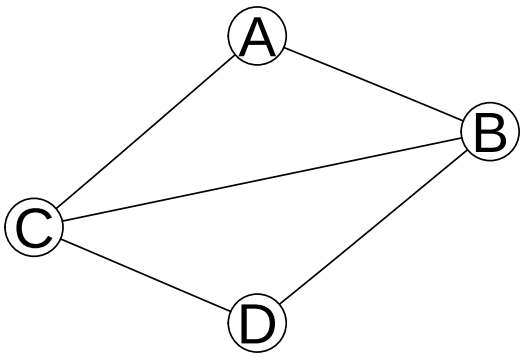
$$L = \prod_x p(x)^{n(x)} = \prod_{A \in \mathcal{A}} \prod_{x_A} \psi_A(x_A)^{n(x_A)}$$

11.3 Graphical models

A hierarchical log–linear model with generating class $\mathcal{A} = \{a_1, \ldots a_Q\}$ is GRAPHICAL if $\mathcal{A}$ are the cliques of the dependence graph.

Example: $\mathcal{A}_1 = \{ABC, BCD\}$ is graphical but $\mathcal{A}_2 = \{AB, AC, BCD\}$ is not graphical. Both have dependence graph with cliques $\mathcal{A}_1$.

```
plot(ug(~A:B:C+B:C:D))
```

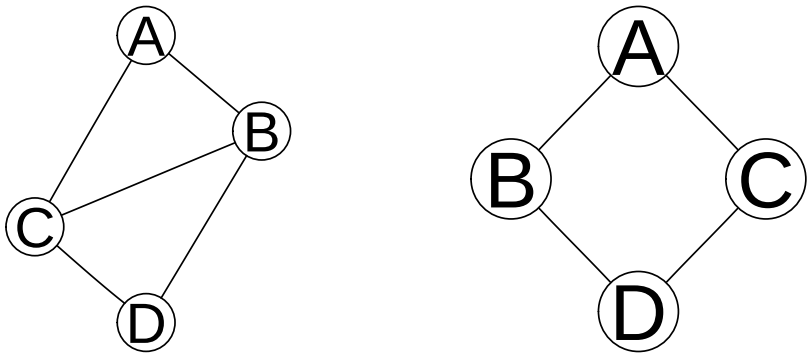


11.4 Decomposable models

A graphical log–linear model is DECOMPOSABLE if the models dependence graph is decomposable.

Example: $\mathcal{A}_1 = \{ABC, BCD\}$ is decomposable but $\mathcal{A}_2 = \{AB, AC, BD, CD\}$ is not.

```
par(mfrow=c(1,2))
plot(ug(~A:B:C+B:C:D))
plot(ug(~A:B+A:C+B:D+C:D))
```



11.5 ML estimation in decomposable models

For a decomposable model, the MLE can be found in closed form as

$$\hat{p}(x) = \frac{\prod_{C:cliques} \hat{p}_C(x_C)}{\prod_{S:separators} \hat{p}_S(x_S)} \tag{3}$$

where $\hat{p}_E(x_E) = n(x_E)/n$ for any clique or separator E.

Consider the lizard data and the model $\mathcal{A} = \{[DS][HS]\}$. The MLE is

$$\hat{p}_{dhs} = \frac{(n_{ds}/n)(n_{hs}/n)}{n_s/n} = \frac{n_{ds}n_{hs}}{nn_s}$$

- The result (3) is IMPORTANT in connection with Bayesian networks, because we obtain a CLIQUE POTENTIAL representation of p directly.
- Hence if we find a decomposable graphical model then we can convert this to a Bayesian network.

---

12 Testing for conditional independence

Tests of general conditional independence hypotheses of the form $u \perp\!\!\!\perp v|W$ can be performed with **ciTest()** (a wrapper for calling **ciTest_table()**).

```
args(ciTest_table)
```

```
function (x, set = NULL, statistic = "dev", method = "chisq",
    adjust.df = TRUE, slice.info = TRUE, L = 20, B = 200, ...)
NULL
```

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
```

The `set` argument can be given in different forms:

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))
```

12.1 What is a CI-test – stratification

Conditional independence of u and v given W means independence of u and v for each configuration w^* of W.

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

Conceptually form a factor S by crossing the factors in W. The test can then be formulated as 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.

```
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"    "adjust.df"
[7] "varNames"  "slice"
```

```
cit$slice
```

```
  statistic   p.value df species
1 0.1779795 0.6731154  1   anoli
2 1.8476671 0.1740550  1    dist
```

The sth slice is a $|u| \times |v|$–table $\{n_{ijs}\}_{i=1...|u|,j=1...|v|}$. 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.

13 Log–linear models using the gRim package

```
data(wine, package="gRbase")
head(wine,4)
```

```
  Cult  Alch Mlca  Ash Aloa Mgns Ttlp Flvn Nnfp Prnt Clri  Hue Oodw Prln
1   v1 14.23 1.71 2.43 15.6  127 2.80 3.06 0.28 2.29 5.64 1.04 3.92 1065
2   v1 13.20 1.78 2.14 11.2  100 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050
3   v1 13.16 2.36 2.67 18.6  101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
4   v1 14.37 1.95 2.50 16.8  113 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480
```

```
dim(wine)
```

```
[1] 178  14
```

`Cult` is grape variety (3 levels); all other variables are results of chemical analyses.

Comes from UCI Machine Learning Repository. "Task" is to predict `Cult` from chemical measurements.

Discretize data:

```
wine <- cbind(Cult=wine[,1],
              as.data.frame(lapply(wine[-1], cut, 2, labels=c('L','H'))))
head(wine)
```

```
  Cult Alch Mlca Ash Aloa Mgns Ttlp Flvn Nnfp Prnt Clri Hue Oodw Prln
1   v1    H    L   H    L    H    H    H    L    H    L   L    H    H
2   v1    H    L   L    L    L    H    H    L    L    L   L    H    H
3   v1    H    L   H    L    L    H    H    L    H    L   L    H    H
4   v1    H    L   H    L    L    H    H    L    H    H   L    H    H
5   v1    H    L   H    H    H    H    L    L    L    L   L    H    L
6   v1    H    L   H    L    L    H    H    L    L    L   L    H    H
```

```
dim(xtabs(~.,wine))
```

```
 [1] 3 2 2 2 2 2 2 2 2 2 2 2 2
```

Just look at some variables

```
wine <- wine[,1:4]
head(wine)
```

```
  Cult Alch Mlca Ash
1   v1    H    L   H
2   v1    H    L   L
3   v1    H    L   H
4   v1    H    L   H
5   v1    H    L   H
6   v1    H    L   H
```

```
dim(xtabs(~.,wine))
```

```
[1] 3 2 2 2
```

The function **dmod()** is used for specifying log–linear models.

- Data must be a table or dataframe (which can be coerced to a table)
- Model given as generating class:
  - A right–hand–sided formula or
  - A list.
  - Variable names may be abbreviated:

```
mm <- dmod(~Cult:Alch+Alch:Mlca:Ash, data=wine)
mm <- dmod(list(c("Cult","Alch"), c("Alch","Mlca","Ash")), data=wine)
mm <- dmod(~C:Alc+Alc:M:As, data=wine)
mm
```

```
Model: A dModel with 4 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :         926.33 mdim :   11 aic :       948.33
 ideviance :         127.86 idf  :    6 bic :       983.33
 deviance  :          48.08 df   :   12
```

The GENERATING CLASS as a list is retrieved with **terms()** and as a formula with **formula()** :

```
str(terms(mm))
```

```
List of 2
 $ : chr [1:2] "Cult" "Alch"
 $ : chr [1:3] "Alch" "Mlca" "Ash"
```

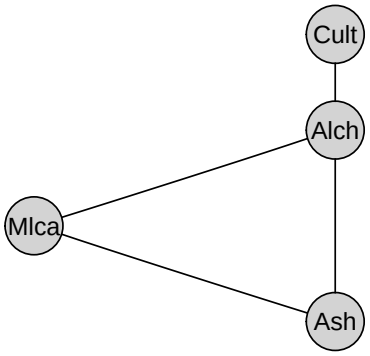
```
formula(mm)
```

```
~Cult * Alch + Alch * Mlca * Ash
```

13.1 Plotting the dependence graph

If `Rgraphviz` is installed, graphs (and models) can be plotted with **plot()** . Otherwise **iplot()** may be used.

```
plot(mm)
```



Notice: No dependence graph in model object; must be generated on the fly using **ugList()** :

```
# Default: a graphNEL object
DG <- ugList(terms(mm))
DG
```

```
A graphNEL graph with undirected edges
Number of Nodes = 4
Number of Edges = 4
```

```
# Alternative: an adjacency matrix
ugList(terms(mm), result="matrix")
```

```
     Cult Alch Mlca Ash
Cult    0    1    0   0
Alch    1    0    1   1
Mlca    0    1    0   1
Ash     0    1    1   0
```

13.2 Model specification shortcuts

Shortcuts for specifying some models

```
str(terms(dmod(~.^., data=wine))) ## Saturated model
```

```
List of 1
 $ : chr [1:4] "Cult" "Alch" "Mlca" "Ash"
```

```
str(terms(dmod(~.^1, data=wine))) ## Independence model
```

```
List of 4
 $ : chr "Cult"
 $ : chr "Alch"
 $ : chr "Mlca"
 $ : chr "Ash"
```

```
str(terms(dmod(~.^3, data=wine))) ## All 3-factor model
```

```
List of 4
 $ : chr [1:3] "Cult" "Alch" "Mlca"
 $ : chr [1:3] "Cult" "Alch" "Ash"
 $ : chr [1:3] "Cult" "Mlca" "Ash"
 $ : chr [1:3] "Alch" "Mlca" "Ash"
```

Useful to combine with specification of a marginal table:

```
marg <- c("Cult", "Alch", "Mlca")
str(terms(dmod(~.^., data=wine, margin=marg))) ## Saturated model
```

```
List of 1
 $ : chr [1:3] "Cult" "Alch" "Mlca"
```

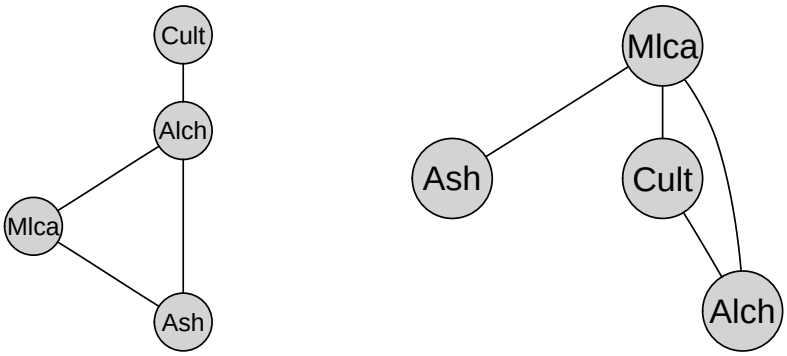
```
str(terms(dmod(~.^1, data=wine, margin=marg))) ## Independence model
```

```
List of 3
 $ : chr "Cult"
 $ : chr "Alch"
 $ : chr "Mlca"
```

13.3 Altering graphical models

Natural operations on graphical models: add and delete edges

```
mm <- dmod(~Cult:Alch+Alch:Mlca:Ash, data=wine)
mm2 <- update(mm, list(dedge=~Alch:Ash, aedge=~Cult:Mlca)) # No abbreviations
par(mfrow=c(1,2)); plot(mm); plot(mm2)
```



13.4 Model comparison

Models are compared with **compareModels()** .

```
mm <- dmod(~Cult:Alch+Alch:Mlca:Ash, data=wine)
mm2 <- update(mm, list(dedge=~Alch:Ash+Alch:Cult)) # No abbreviations
compareModels(mm,mm2)
```

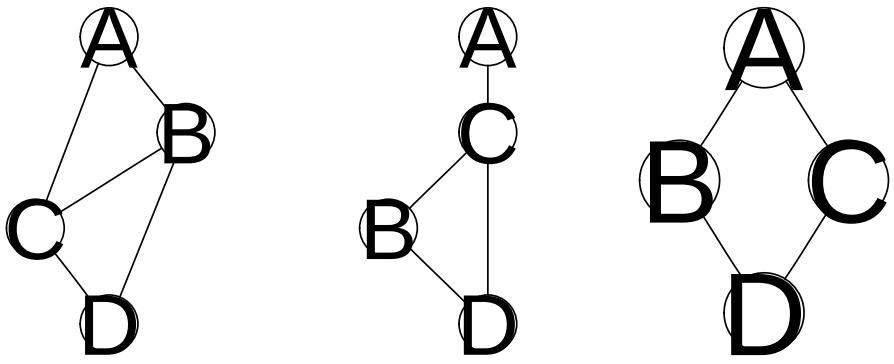
```
Large:
  :"Cult" "Alch"
```

```
  :"Alch" "Mlca" "Ash"
Small:
  :"Alch" "Mlca"
  :"Mlca" "Ash"
  :"Cult"
-2logL:   126.66 df: 4 AIC(k= 2.0):   118.66 p.value: 0.000000
```

13.5 Decomposable models – deleting edges

Result: If $\mathcal{A}_1$ is a decompsable model and we remove an edge $e = \{u, v\}$ which is contained in one clique C only, then the new model $\mathcal{A}_2$ will also be decomposable.

```
par(mfrow=c(1,3))
plot(ug(~A:B:C+B:C:D))
plot(ug(~A:C+B:C+B:C:D))
plot(ug(~A:B+A:C+B:D+C:D))
```



Left: $\mathcal{A}_1$ – decomposable; Center: dropping $\{A, B\}$ gives decomposable model; Right: dropping $\{B, C\}$ gives non–decomposable model.

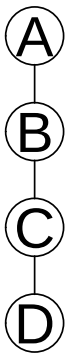
Result: The test for removal of $e = \{u, v\}$ which is contained in one clique C only can be made as a test for $u \perp\!\!\!\perp v | C \setminus \{u, v\}$ in the C–marginal table.

This is done by **ciTest()** . Hence, no model fitting is necessary.

13.6 Decomposable models – adding edges

More tricky when adding edge to a decomposable model

```
plot(ug(~A:B+B:C+C:D))
```



Adding $\{A, D\}$ gives non–decomposable model; adding $\{A, C\}$ gives decomposable model.

One solution: Try adding edge to graph and test if new graph is decomposable. Can be tested with MAXIMUM CARDINALITY SEARCH as implemented in **mcs()** . Runs in $O(|edges| + |vertices|)$.

```
UG <- ug(~A:B+B:C+C:D)
mcs(UG)
```

```
[1] "A" "B" "C" "D"
```

```
UG1 <- addEdge("A","D",UG)
mcs(UG1)
```

```
character(0)
```

```
UG2 <- addEdge("A","C",UG)
mcs(UG2)
```

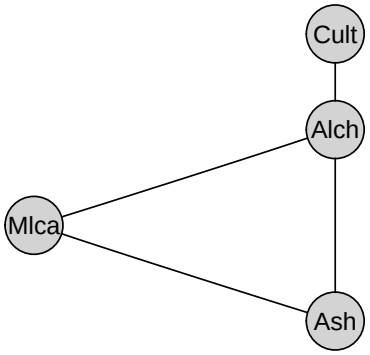
```
[1] "A" "B" "C" "D"
```

13.7 Test for adding and deleting edges

Done with **testdelete()** and **testadd()**

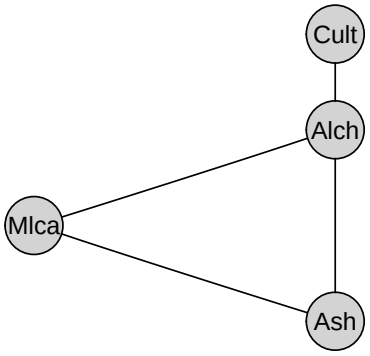
```
mm <- dmod(~C:Alc+Alc:M:As, data=wine)
plot(mm)
testdelete(mm, edge=c("Mlca","Ash"))
```

```
dev:    7.710 df:  2 p.value: 0.02117 AIC(k=2.0):    3.7 edge: Mlca:Ash
host:  Alch Mlca Ash
Notice: Test performed in saturated marginal model
```



```
mm <- dmod(~C:Alc+Alc:M:As, data=wine)
plot(mm)
testadd(mm, edge=c("Mlca","Cult"))
```

```
dev:   29.388 df:  4 p.value: 0.00001 AIC(k=2.0):  -21.4 edge: Mlca:Cult
host:  Alch Mlca Cult
Notice: Test performed in saturated marginal model
```



13.8 Model search in log–linear models using `gRim`

Model selection implemented in **stepwise()** function.

- Backward / forward search (Default: backward)
- Select models based on p–values or AIC(k=2) (Default: AIC(k=2))

- Model types can be ”unsrestricted” or ”decomposable”. (Default is decomposable if initial model is decompsable)
- Search method can be ”all” or ”headlong”. (Default is all)

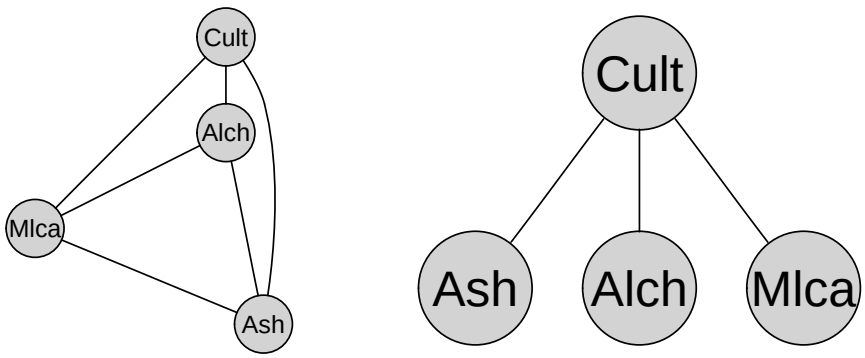
```
args(stepwise.iModel)
```

```
function (object, criterion = "aic", alpha = NULL, type = "decomposable",
    search = "all", steps = 1000, k = 2, direction = "backward",
    fixinMAT = NULL, fixoutMAT = NULL, details = 0, trace = 2,
    ...)
NULL
```

```
dm1 <- dmod(~.^., data=wine)
dm2 <- stepwise(dm1, 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=0.00
. Initial model: is graphical=TRUE is decomposable=TRUE
  change.AIC   -2.5140 Edge deleted: Ash Alch
  change.AIC   -1.7895 Edge deleted: Ash Mlca
  change.AIC   -0.8054 Edge deleted: Mlca Alch
```

```
par(mfrow=c(1,2)); plot(dm1); plot(dm2)
```

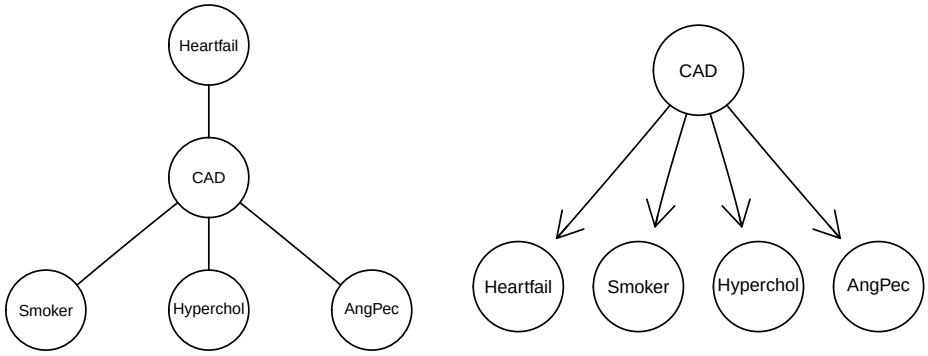


---

14 From graph and data to network

Consider naive Bayesian model for CAD data: All risk factors/symptoms are conditionally independent given the disease:

```
par(mfrow=c(1,2))
plot(UG  <- ug(~Heartfail:CAD+Smoker:CAD+Hyperchol:CAD+AngPec:CAD))
plot(DAG <- dag(~Heartfail:CAD+Smoker:CAD+Hyperchol:CAD+AngPec:CAD))
```



From a statistical point of view these two models are equivalent.

Given either a DAG or an UG and data (either as a table or as a dataframe) we can construct BN's on the fly:

```
cadmod1 <- compile(grain(UG, cad1))
cadmod2 <- compile(grain(DAG, cad1))
```

```
querygrain(cadmod1, nodes="CAD")
```

```
$CAD
CAD
       No       Yes
0.5466102 0.4533898
```

```
querygrain(cadmod2, nodes="CAD")
```

```
$CAD
CAD
       No       Yes
0.5466102 0.4533898
```

14.1 Prediction

We shall try to predict `CAD` in the validation dataset `cad2`

```
data(cad2)
head(cad2,3)
```

```
     Sex AngPec        AMI QWave QWavecode    STcode STchange SuffHeartF
1   Male   None NotCertain    No    Usable    Usable      Yes         Yes
2 Female   None NotCertain    No    Usable    Usable      Yes         Yes
3 Female   None NotCertain    No Nonusable Nonusable       No          No
  Hypertrophi Hyperchol Smoker Inherit Heartfail CAD
1          No        No   <NA>      No        No  No
2          No        No   <NA>      No        No  No
3          No       Yes   <NA>      No        No  No
```

using **predict.grain()** .

```
args(predict.grain)
```

```
function (object, response, predictors = setdiff(names(newdata),
    response), newdata, type = "class", ...)
NULL
```

```
pred1 <- predict(cadmod1, resp="CAD", newdata=cad2, type="class")
str(pred1)
```

```
List of 2
 $ pred    :List of 1
  ..$ CAD: chr [1:67] "No" "No" "No" "No" ...
 $ pFinding: num [1:67] 0.1382 0.1382 0.1102 0.0413 0.1102 ...
```

```
pred2 <- predict(cadmod1, resp="CAD", newdata=cad2, type="dist")
str(pred2)
```

```
List of 2
 $ pred    :List of 1
  ..$ CAD: num [1:67, 1:2] 0.914 0.914 0.679 0.604 0.679 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:2] "No" "Yes"
 $ pFinding: num [1:67] 0.1382 0.1382 0.1102 0.0413 0.1102 ...
```

14.2 Classification error

```
table(cad2$CAD)
```

```
No Yes
41  26
```

```
table(cad2$CAD)/sum(table(cad2$CAD))
```

```
       No       Yes
0.6119403 0.3880597
```

```
tt <- table(cad2$CAD, pred1$pred$CAD)
tt
```

```
     No Yes
  No  34   7
  Yes 14  12
```

```
sweep(tt, 1, apply(tt,1,sum), FUN="/")
```

```
            No       Yes
  No  0.8292683 0.1707317
  Yes 0.5384615 0.4615385
```

15 Winding up – and practicals

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 `gRbase` 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.

Practical:

- Find a decomposable graphical model for the coronary artery disease data `cad1`. Create a Bayesian network from this model.
- Predict the disease variable `CAD` in the validation dataset `cad2` (which has incomplete observations).
- How good prediction results can you obtain?

Index