Graphical Models with R

Tutorial at UiO, Norway, November 2012

Søren Højsgaard

Department of Mathematical Sciences

Aalborg University, Denmark

November 25, 2012

Printed: November 25, 2012 File: GMwR-slides.tex

Contents

1 Outline of tutorial

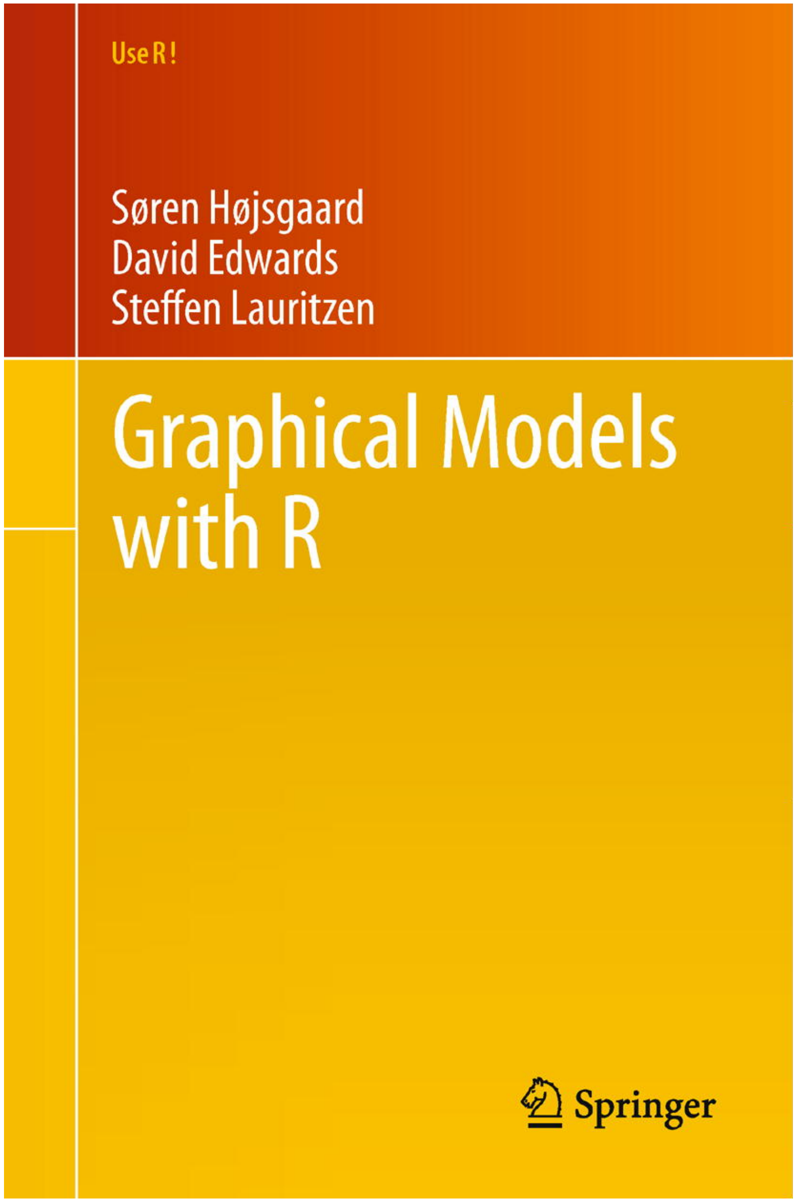
- Theoretical part: 3 lectures, each about 45 minutes.
- Practical part: 3 hours of computer exercises.

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

1.1 Topics

- Bayesian networks and 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.

1.2 Book: Graphical Models with R



Use R!

Søren Højsgaard
David Edwards
Steffen Lauritzen

Graphical Models with R

Springer

1.3 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.

1.4 The practicals: The coronary artery disease data

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

```
R> data(cad1)
R> 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.

```
R> data(cad2)
R> 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
```

1.5 Graphical models in a few words

- The "language" of graphical models is *conditional independence* restrictions among variables.
- Used for identifying *direct associations* and *indirect associations* among random variables.
- Used for breaking a large complex stochastic model into smaller components.
- Used for very efficient calculation of conditional distributions via message passing.
- Graphs provide tool for interpreting models
- Graphs provide terminology for organizing certain computations

2 Conditional independence

Let X, Y be random variables. X and Y are independent if

$$f_{X,Y}(x,y) = f_X(x)f_Y(y)$$

or, equivalently, if

$$f_{Y|X}(y|x) = f_Y(y)$$

Let X, Y, Z be random variables. X and Y are conditionally independent given Z (written $X \perp\!\!\!\perp Y|Z$) if for each value z of Z, X and Y are independent in the conditional distribution given $Z = z$. That is if

$$f_{X,Y|Z}(x,y|z) = f_{X|Z}(x|z)f_{Y|Z}(y|z)$$

– or equivalently

$$f_{Y|X,Z}(y|x,z) = f_{Y|Z}(y|z)$$

So if $Z = z$ is known then knowledge of X will provide no additional knowledge of Y.

2.1 Factorization criterion

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 $h()$.

Example 2.1

$$X = (X_1, X_2, X_3)^\top \sim N_3(0, \Sigma), \quad \Sigma^{-1} = K = \begin{bmatrix} k_{11} & k_{12} & 0 \\ k_{21} & k_{22} & k_{23} \\ 0 & k_{32} & k_{33} \end{bmatrix}$$

Then $X_1 \perp\!\!\!\perp X_3 \,|\, X_2$ because $f(x) \propto \exp(x^\top K x)$ becomes

$$\begin{aligned} f(x_1, x_2, x_3) &\propto \exp\left(x_1^2 k_{11} + 2x_1 x_2 k_{12} + x_2^2 k_{22} + 2x_2 x_3 k_{23} + x_3^2 k_{33}\right) \\ &= g(x_1, x_2)h(x_2, x_3) \end{aligned}$$

□

Example 2.2 Let X_1, X_2, X_3 be discrete with

$$p_{ijk} = P(X_1 = i, X_2 = j, X_3 = k)$$

In a log–linear model we may have, for example,

$$\log p_{ijk} = \alpha_i^1 + \alpha_j^2 + \alpha_k^3 + \beta_{ij}^{12} + \beta_{jk}^{23}$$

Exponentiating and collecting terms gives

$$p_{ijk} = g(i,j)h(j,k)$$

Hence $X_1 \perp\!\!\!\perp X_3 \mid X_2$. □

3 Undirected Graphs

Definition 1 *An (undirected) graph as a mathematical object is a pair $\mathcal{G} = (V, E)$ where V is a set of vertices (or nodes) and E is a set of edges (and edge is a pair of vertices).*

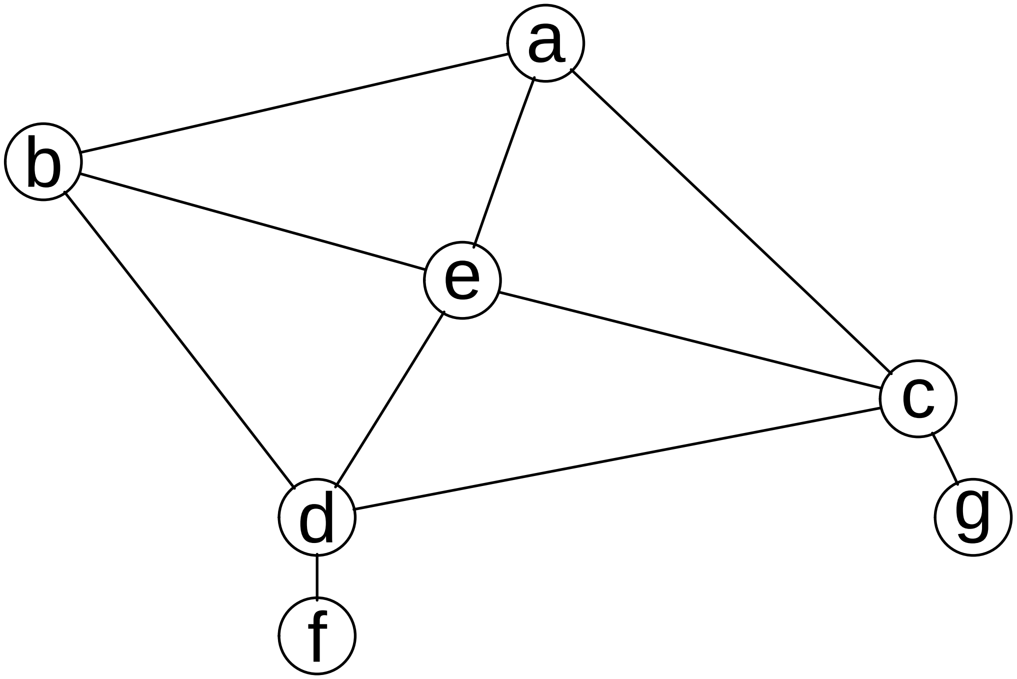
Definition 2 *Given a set of vertices V, a collection $\mathcal{A} = \{a_1, \ldots, a_Q\}$ of subsets of V where $\cup_j a_j = V$. The graph* generated *by $\mathcal{A}$ is $\mathcal{G}(\mathcal{A}) = (V, E)$ where E is given as follows: $\{\alpha, \beta\} \in E$ iff $\{\alpha, \beta\} \subset a_j$ for some j.*

The function *ug()* from **gRbase** creates an undirected graph:

```
R> library(gRbase)
R> g1 <- ug(~a:b:e + a:c:e + b:d:e + c:d:e + c:g + d:f)
R> class(g1)

[1] "graphNEL"
attr(,"package")
[1] "graph"

R> as(g1, "matrix")

  a b e c d g f
a 0 1 1 1 0 0 0
b 1 0 1 0 1 0 0
e 1 1 0 1 1 0 0
c 1 0 1 0 1 1 0
d 0 1 1 1 0 0 1
g 0 0 0 1 0 0 0
f 0 0 0 0 1 0 0
```

Graphs can be displayed with the *plot()* method

```
R> library(Rgraphviz)
R> plot(g1)
```

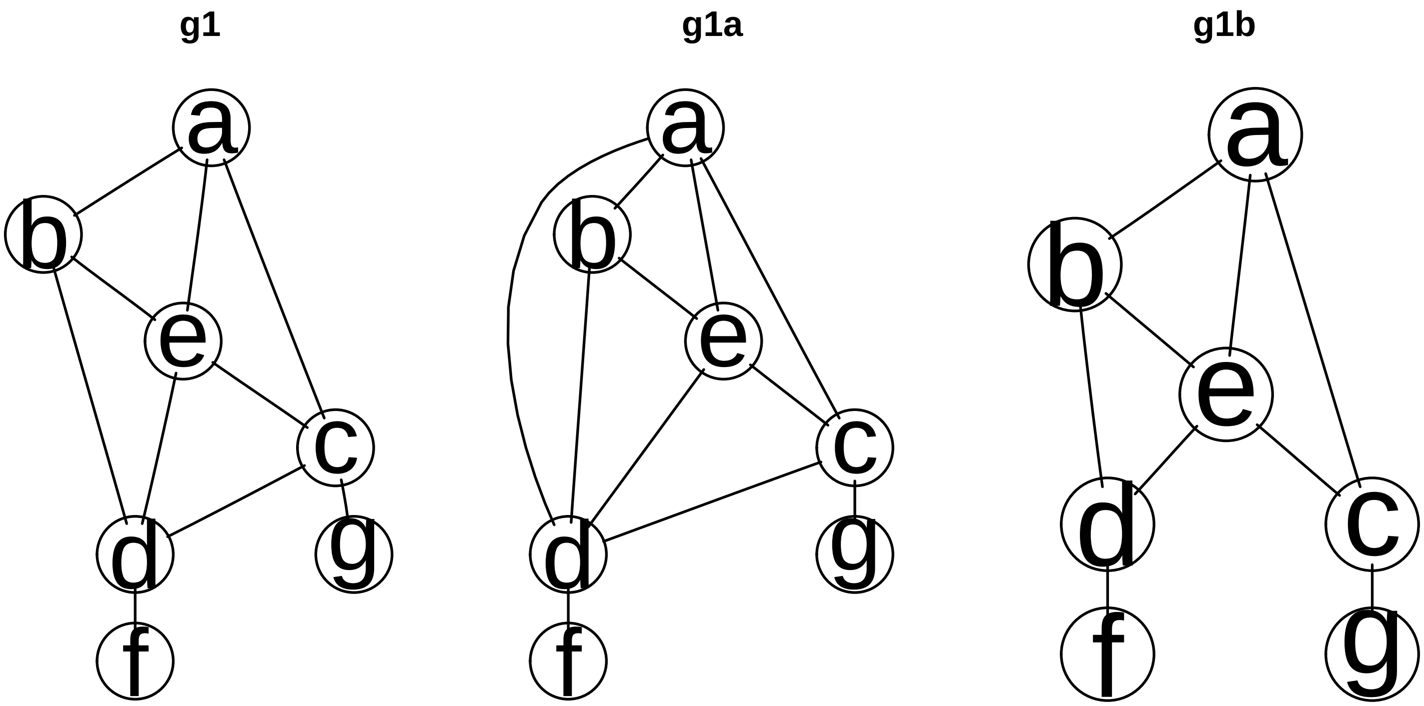


Graphs can also be defined using adjacency matrices:

```
R> m <- matrix(c(0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
                 1, 1, 0, 1, 1, 1, 1, 1, 0), nrow = 5)
R> rownames(m) <- colnames(m) <- c("a", "b", "c", "d", "e")
R> m

  a b c d e
a 0 1 1 0 1
b 1 0 0 1 1
c 1 0 0 1 1
d 0 1 1 0 1
e 1 1 1 1 0

R> as(m, "graphNEL")

A graphNEL graph with undirected edges
Number of Nodes = 5
Number of Edges = 8
```

Graphs can be altered using *addEdge()* and *removeEdge()*

```
R> g1a <- addEdge("a", "d", g1)
R> g1b <- removeEdge("c", "d", g1)
R> par(mfrow = c(1, 3))
R> plot(g1, main = "g1")
R> plot(g1a, main = "g1a")
R> plot(g1b, main = "g1b")
```

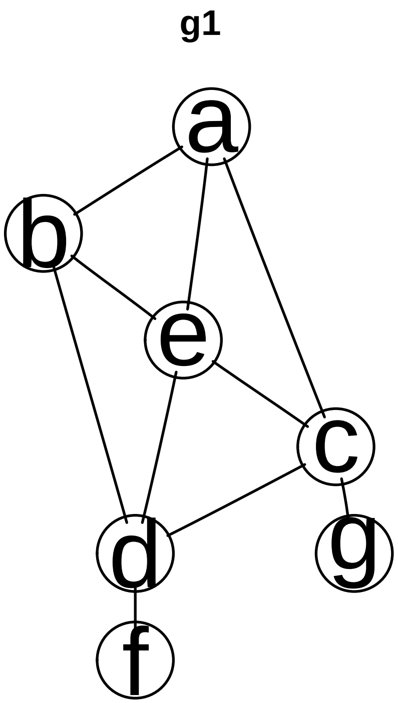


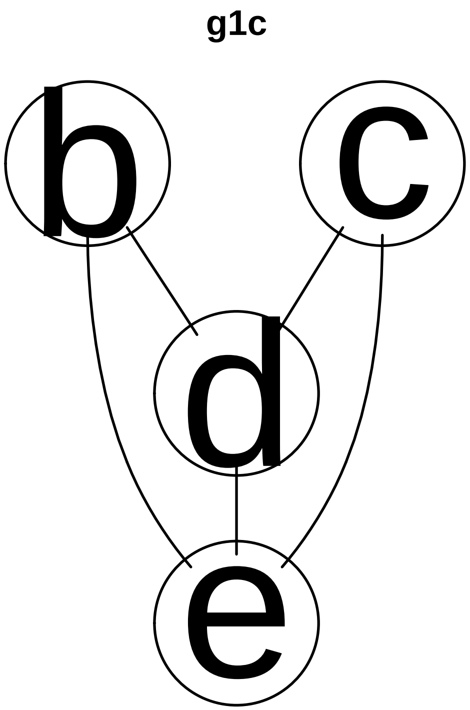
Definition 3 *The graph* $\mathcal{G}_0 = (V_0; E_0)$ *is said to be a* subgraph *of* $\mathcal{G} = (V, E)$ *if* $V_0 \subset V$ *and* $E_0 \subset E$*.*

For $A \subset V$, *let* E_A *denote the set of edges in* E *between vertices in* A*. Then* $\mathcal{G}_A = (A, E_A)$ *is the* subgraph induced by A*.*

For example

```
R> g1c <- subGraph(c("b", "c", "d", "e"), g1)
R> par(mfrow = c(1, 3))
R> plot(g1, main = "g1")
R> plot(g1c, main = "g1c")
```


g1
a
b
e
c
d
g
f


g1c
b
c
d
e

Definition 4 *A set $A \subset V$ of vertices in a graph $\mathcal{G} = (V, E)$ is* <u>complete</u> *if all pairs of vertices in A are connected by an edge. A graph $\mathcal{G} = (V, E)$ is* <u>complete</u> *if V is complete.*

Definition 5 *A* <u>clique</u> *is a maximal complete subset, that is a complete subset which is not contained in a larger complete subset.*

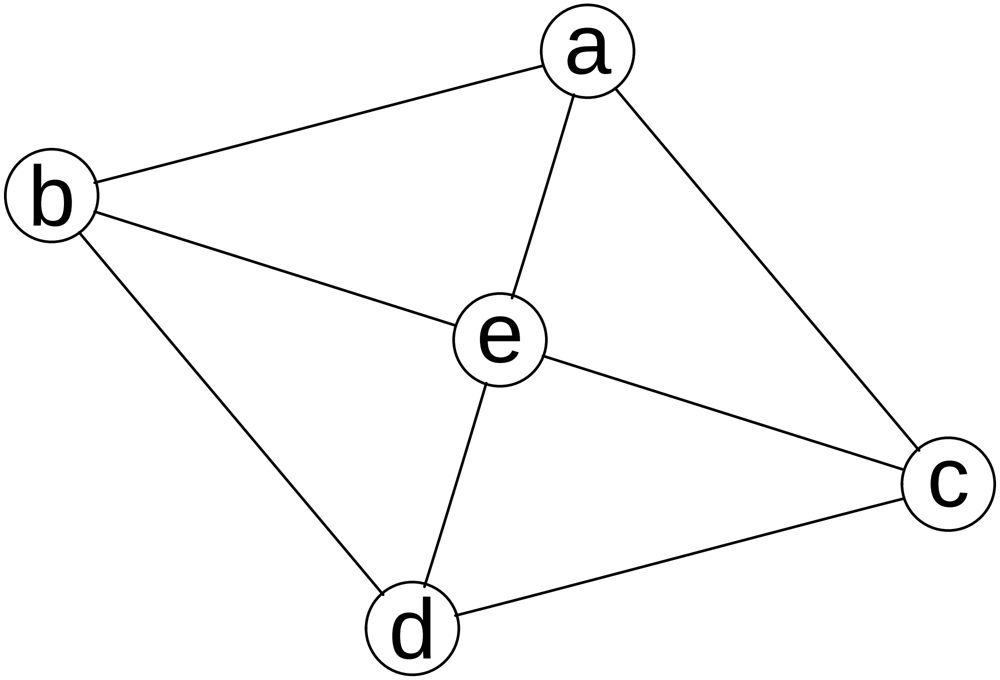
```
R> is.complete(g1,set=c("a","b","e"))

[1] TRUE

R> is.complete(g1)

[1] FALSE

R> str(maxClique(g1))

List of 1
 $ maxCliques:List of 6
  ..$ : chr [1:3] "e" "b" "a"
  ..$ : chr [1:3] "e" "b" "d"
  ..$ : chr [1:3] "e" "c" "a"
  ..$ : chr [1:3] "e" "c" "d"
  ..$ : chr [1:2] "g" "c"
  ..$ : chr [1:2] "f" "d"
```

Definition 6 *A* path *(of* length *n) between α and β in an undirected graph is a set of vertices $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n = \beta$ where $\{\alpha_{i-1}, \alpha_i\} \in E$ for $i = 1, \ldots, n$.*

If a path has $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n = \beta$ has $\alpha = \beta$ then the path is said to be a cycle *(of length n).*

Definition 7 *A subset $S \subset V$ is said to* separate *$A \subset V$ and $B \subset V$ if every path between a vertex in A and a vertex in B contains a vertex from S.*

```
R> g2 <- ug(~a:b:e + a:c:e + b:d:e + c:d:e)
R> plot(g2)
R> separates("a", "d", c("b", "c", "e"), g2)

[1] TRUE
```



3.1 Factorization and dependence graph

Consider d–dimensional random vector $X = (X_i; i \in V)$ where $V = \{1, \dots, d\}$.

For $a \subset V$ define $X_a = (X_i; i \in a)$.

Let $\mathcal{A} = \{a_1, \dots, a_Q\}$ be a collection of subset of V where $\cup_j a_j = V$.

Consider pmf's/pdf's of the form

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

where $\phi_a()$ is a non–negative function of x_a (equivalently: a function that depends in x only through x_a).

We shall often just write

$$p = \prod_{a \in \mathcal{A}} \phi_a \text{ or } p = \prod_{a \in \mathcal{A}} \phi(a)$$

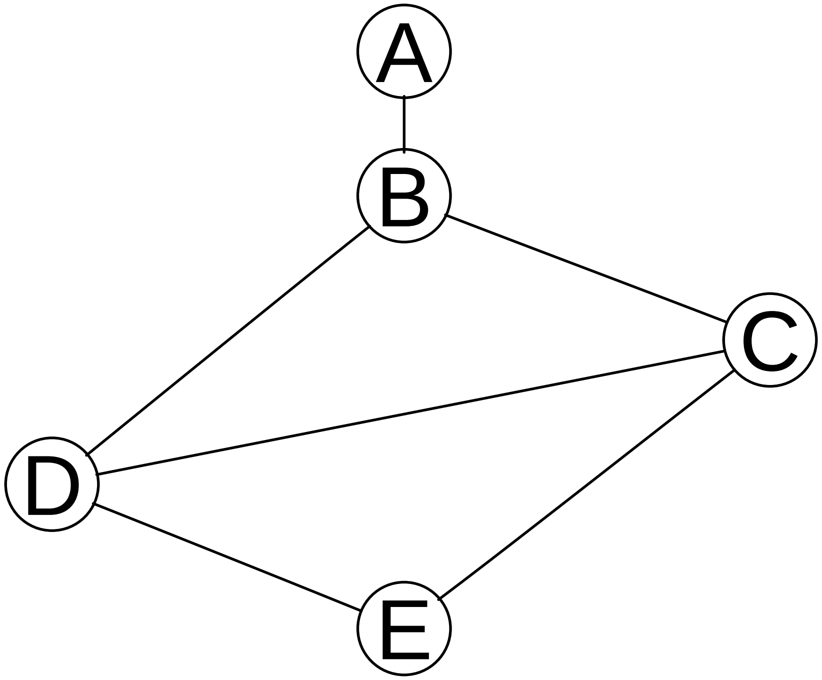
The dependence graph for p is the graph induced by $\mathcal{A}$.

Suppose

$$p(x) = \psi_{AB}(x_{AB})\psi_{BCD}(x_{BCD})\psi_{CE}(x_{CE})\psi_{DE}(x_{DE})$$

Then the *dependence graph* for p is:

```
R> plot((g3 <- ug(~ A:B + B:C:D + C:E + D:E)))
```



3.2 Reading conditional independencies – global Markov property

Conditional independencies can be read off the dependence graph:

- *Global Markov Property* : If $A \subset V$ and $B \subset V$ are separated by $S \subset V$ in the dependence graph $\mathcal{G}$ then $X_A \perp\!\!\!\perp A_B | X_S$.
- Follows from factorization criterion: $X \perp\!\!\!\perp Y | Z$ if

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

- Example: With

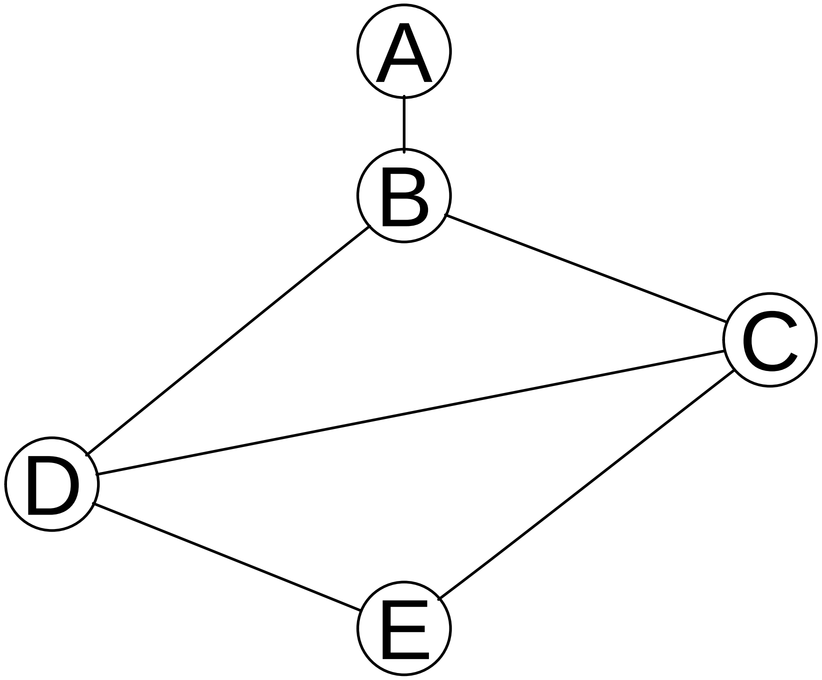
$$p(x) = \psi_{AB}(x_{AB}) \psi_{BCD}(x_{BCD}) \psi_{CE}(x_{CE}) \psi_{DE}(x_{DE})$$

we have $(D, E) \perp\!\!\!\perp A | B$:

$$\begin{aligned} p(x) &= \Big[\psi_{AB}(x_{AB})\Big]\Big(\psi_{BCD}(x_{BCD}) \psi_{CE}(x_{CE}) \psi_{DE}(x_{DE})\Big) \\ &= g(x_{AB}) h(x_{BCDE}) \end{aligned}$$

Now integrate over x_C and the result follows.

```
R> plot(g3)
R> separates(c("D","E"), "A", "B", g3)

[1] TRUE
```



4 Directed acyclic graphs (DAGs)

Definition 8 *A* directed graph *as a mathematical object is a pair* $\mathcal{G} = (V, E)$ *where* V *is a set of vertices and* E *is a set of directed edges, normally drawn as arrows.*

A directed graph is acyclic *if it has no directed cycles, that is, cycles with the arrows pointing in the same direction all the way around.*

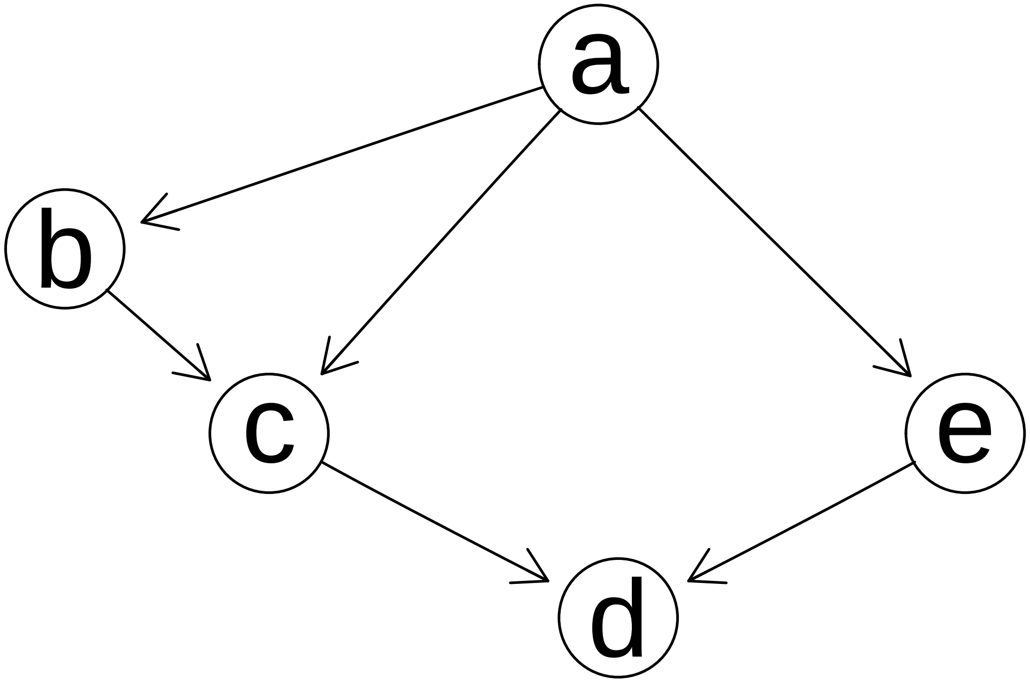
A DAG *is a directed graph that is acyclic.*

A DAG is created by the *dag()* function in **gRbase**. The graph can be specified by a list of formulas or by a list of vectors. The following statements are equivalent:

```
R> dag0 <- dag(~a, ~b * a, ~c * a * b, ~d * c * e, ~e * a)
R> dag0 <- dag(~a + b:a + c:a:b + d:c:e + e:a)
R> dag0 <- dag("a", c("b", "a"), c("c", "a", "b"), c("d", "c", "e"), c("e", "a"))
R> dag0

A graphNEL graph with directed edges
Number of Nodes = 5
Number of Edges = 6
```

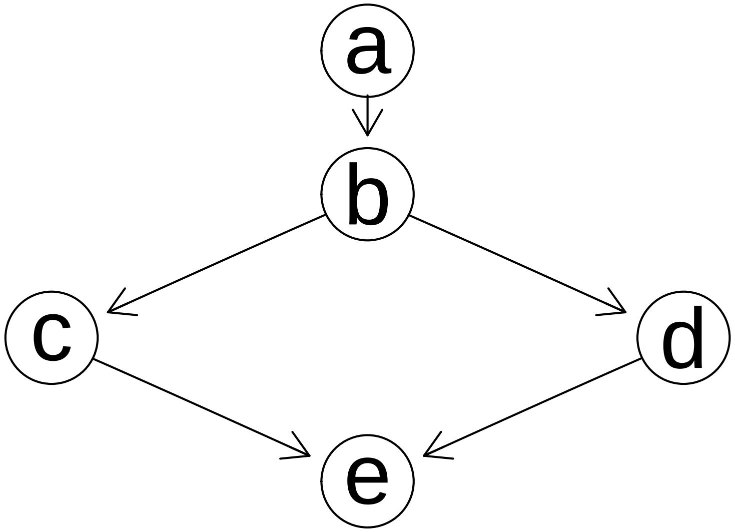
Note that `d*b*c` and `d:b:c` means that ”d” has parents ”b” and ”c”.

```
R> plot(dag0)
```



Directed graphs can also be created from matrices:

```
R> (m <- matrix(c(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
                  1, 0, 0, 0, 0, 0, 1, 1, 0), nrow = 5))

     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    0    0    0
[2,]    0    0    1    1    0
[3,]    0    0    0    0    1
[4,]    0    0    0    0    1
[5,]    0    0    0    0    0

R> rownames(m) <- colnames(m) <- letters[1:5]
R> dg <- as(m, "graphNEL")
R> plot(dg)
```



Definition 9 *The parents of a vertex β are those nodes α for which $\alpha \to \beta$. Denote this set $\mathrm{pa}(\beta)$.*

The children of α are those nodes β for which $\alpha \to \beta$. Denote this set by $\mathrm{ch}(\alpha)$.

```
R> parents("d", dag0)

[1] "c" "e"

R> children("c", dag0)

[1] "d"
```

4.1 Factorization and dependence graph – DAGs

Let $\mathcal{D} = (V, E)$ be a DAG, $X = (X_v; v \in V)$ be random vector with density $p()$. If $p()$ factorizes as

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

then p factorizes according to $\mathcal{D}$.

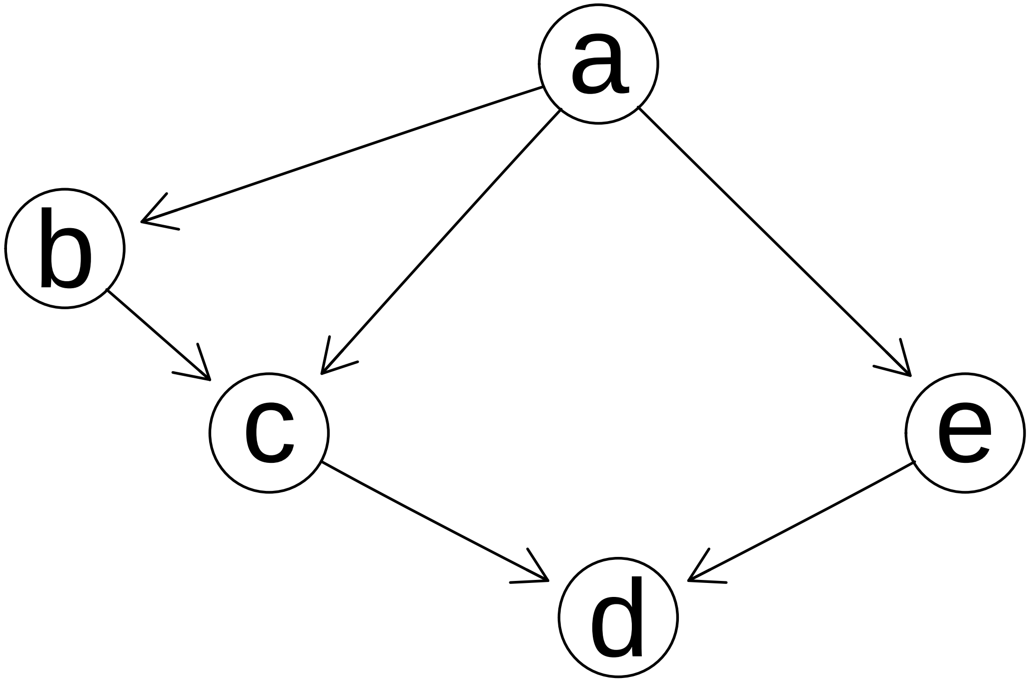
Often just write

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

or

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

```
R> plot(dag0)
```



Factorization for $X = (X_a, X_b, \dots, X_e)$

$$p_X(x) = p_a(x_a) p_{b|a}(x_b|x_a) p_{c|a,b}(x_c|x_a, x_b) p_{e|a}(x_e|x_a) p_{d|c,e}(x_d|x_c, x_e)$$

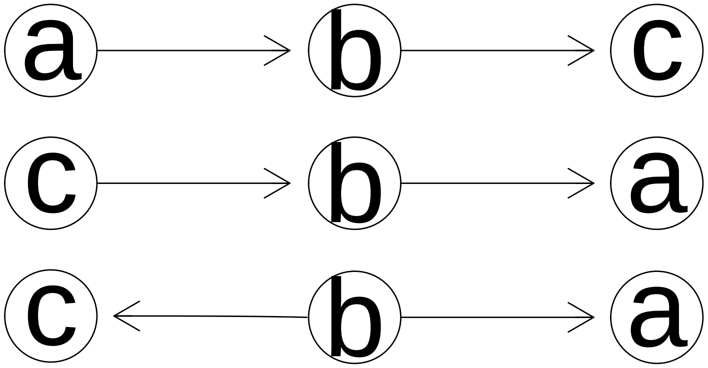
In short form

$$p_V(V) = p(a)p(b|a)p(c|a,b)p(e|a)p(d|c,e)$$

4.2 Reading conditional independencies from DAGs (I)

Reading conditional independencies is different:

```
R> par(mfrow=c(3,1),mar=c(3,1,2,1))
R> plot(dag(~a+b:a+c:b),"circo")
R> plot(dag(~c+b:c+a:b),"circo")
R> plot(dag(~b+a:b+c:b),"circo")
```

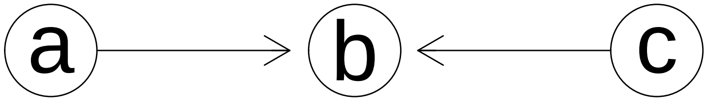


In all cases $a \perp\!\!\!\perp c \mid b$:

$$\begin{aligned} p(a)p(b|a)p(c|b) &= \psi_1(a,b)\psi_2(b,c) \\ p(c)p(b|c)p(a|b) &= \psi_1(a,b)\psi_2(b,c) \\ p(b)p(c|b)p(a|b) &= \psi_1(a,b)\psi_2(b,c) \end{aligned}$$

But this one is different:

```
R> plot(dag(~a+c+b:a:c),"circo")
```



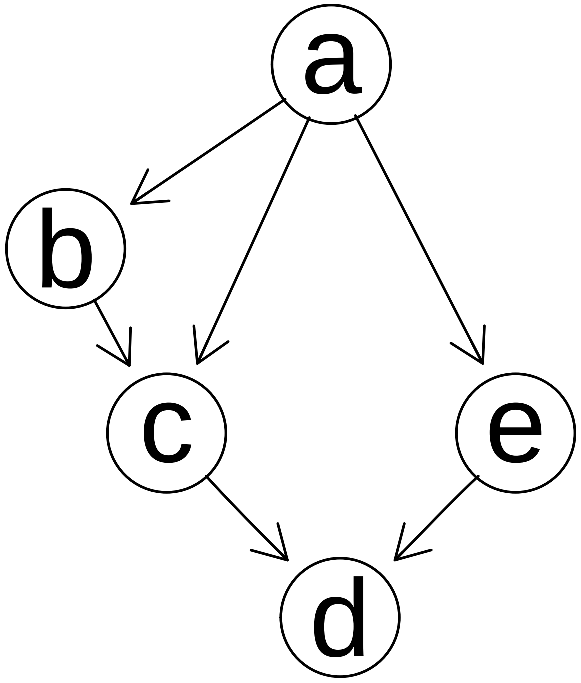
$$p(a)p(c)p(b|a,c)$$

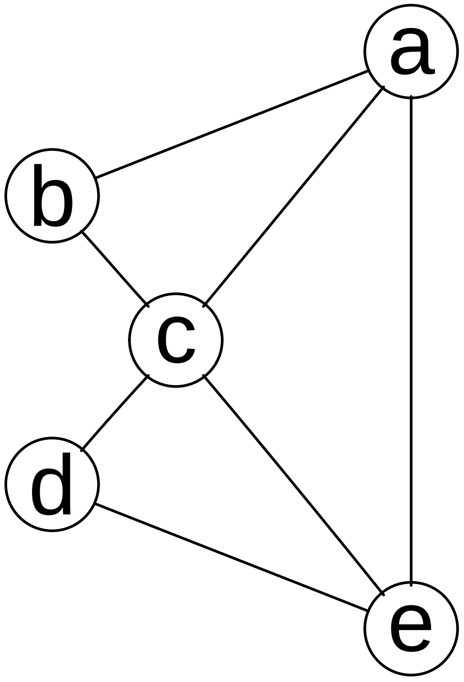
- No factorization so no conditional independence.
- But marginally, $p(a,b) = p(a)p(b)$ so $a \perp\!\!\!\perp b$.

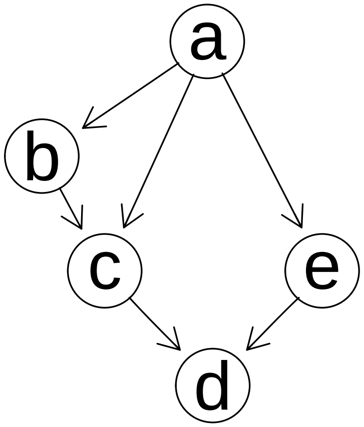
4.3 Moralization

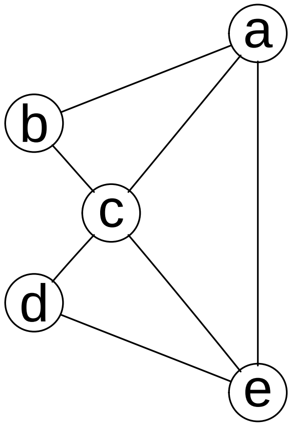
An important operation on DAGs is to (i) add edges between the parents of each node, and then (ii) replace all directed edges with undirected ones, thus returning an undirected graph. This is known as moralization.

```
R> dag0m <- moralize(dag0)
R> par(mfrow=c(1,2))
R> plot(dag0)
R> plot(dag0m)
```









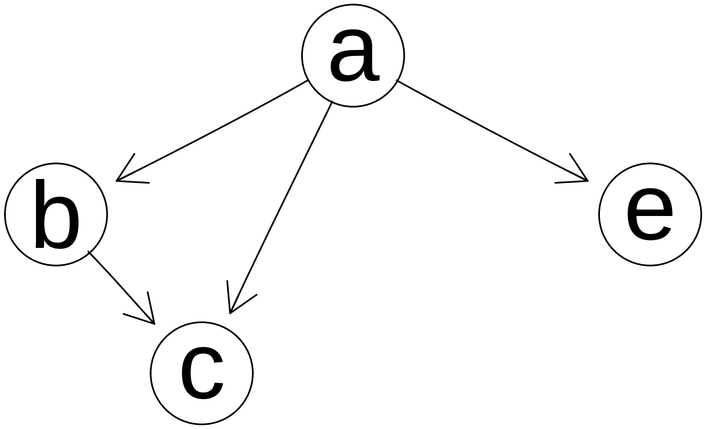
$$\begin{aligned} p(V) &= \Big(p(a)p(b|a)p(c|a,b)\Big)p(e|a)p(d|c,e) \\ &= \psi(c,a,b)\psi(e,a)\psi(d,c,e) = \psi(c,a,b)\psi(c,e,a)\psi(d,c,e) \end{aligned}$$

- Hence, if p factorizes according to DAG then p also factorizes according to the moral graph.
- Therefore the conditional indpendencies that can be read of the moral graph holds – but there may be more: For example: $c \perp\!\!\!\perp e \,|\, a$ – but this can not be seen from moral graph.

4.4 Ancestral sets and graphs*

Definition 10 *If there is a path from α to β we write $\alpha \mapsto \beta$. The ancestors of a node β are the nodes α such that $\alpha \mapsto \beta$. The* ancestral set *of a set A is the union of A with its ancestors. The* ancestral graph *of a set A is the subgraph induced by the ancestral set of A.*

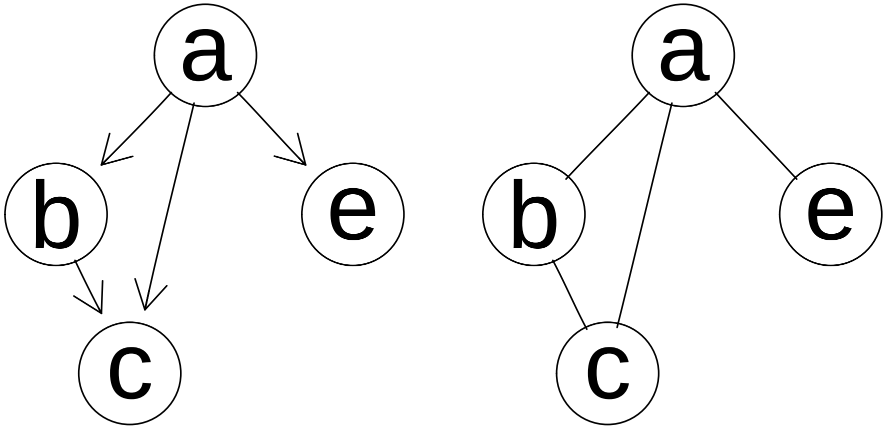
```
R> ancestralSet(c("a", "c", "e"), dag0)

[1] "a" "b" "c" "e"

R> plot(ancestralGraph(c("a", "c", "e"), dag0))
```



4.5 Reading conditional independences from DAG (II)*

To check if $A \perp\!\!\!\perp B|S$ form the ancestral graph of $A \cup B \cup S$. Moralize this ancestral graph. If A and B are separated by S in this moral graph then $A \perp\!\!\!\perp B|S$.

```
R> par(mfrow=c(1,2))
R> plot(ancestralGraph(c("a", "c", "e"), dag0))
R> plot(moralize(ancestralGraph(c("a", "c", "e"), dag0)))
```



Why this works: Because we can integrate over the variables not in the ancestral set of $A \cup B \cup S$. Then we use the factorization structure in $p(\mathrm{An}(A \cup B \cup S))$.

5 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.

  This is the main point when we come to linking BNs to statistical models and data!!!

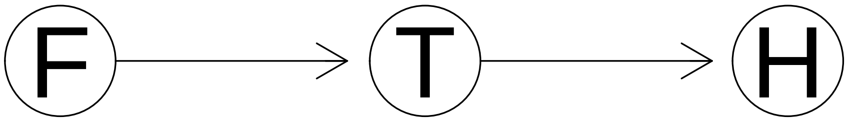
6 A small worked example BN

Consider the following narrative:

> Having *flu (F)* may cause elevated *temperature (T)*. Elevated tempearture may cause a *headache (H)*.

Illustrate this narrative by *DAG* :

```
R> plot((FTH<-dag(~ F + T:F + H:T)), "circo")
```



We define a joint pmf for X as

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

In a less rigorous notation (1) may be written

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

Conditional independence assumption: headache is *conditionally independent* of flu given temperature.

Conditional independence assumption: Flu does not directly cause headache; the headache comes from the fever.

Given a *finding* or *evidence* that a person has headache we may want to calculate

$$P(F = yes|H = yes) \text{ or } P(T = yes|H = yes)$$

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

6.1 Specification of conditional probability tables

We specify $p(F)$, $p(T|G)$ and $p(H|T)$ as tables ($1 =$yes, $2 =$no):

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

F
  F1   F2
0.01 0.99

R> (p.TgF <- parray(c("T","F"), levels=c(2,2), values=c(.95,.05, .001,.999)))

    F
T      F1    F2
  T1 0.95 0.001
  T2 0.05 0.999

R> (p.HgT <- parray(c("H","T"), levels=c(2,2), values=c(.80,.20, .010,.990)))

    T
H     T1   T2
  H1 0.8 0.01
  H2 0.2 0.99
```

6.2 Brute force computations

1) Calculate joint distribution $p(FTH)$

```
R> p.FT  <- tableMult(p.F, p.TgF)
R> p.FTH <- tableMult(p.FT, p.HgT)
R> as.data.frame.table(p.FTH)

   H  T  F      Freq
1 H1 T1 F1 0.0076000
2 H2 T1 F1 0.0019000
3 H1 T2 F1 0.0000050
4 H2 T2 F1 0.0004950
5 H1 T1 F2 0.0007920
6 H2 T1 F2 0.0001980
7 H1 T2 F2 0.0098901
8 H2 T2 F2 0.9791199
```

2) Calculate the marginal distribution $p(FH)$

```
R> p.FH <- tableMargin(p.FTH, margin=c('F','H'))
R> as.data.frame.table(p.FH)

   F  H      Freq
1 F1 H1 0.0076050
2 F2 H1 0.0106821
3 F1 H2 0.0023950
4 F2 H2 0.9793179
```

3) calculate conditional distribution $p(I|H)$

```
R> p.H   <- tableMargin(p.FH, margin='H')
R> (p.FgH <- tableDiv(p.FH, p.H))

    F
H             F1        F2
  H1 0.415866923 0.5841331
  H2 0.002439613 0.9975604
```

So $p(F = 1(yes)|H = 1(yes)) = 0.42$ while $p(F = 1(yes)) = 0.01$.

6.3 Brute force computations will fail

However, this scheme is computationally prohibitive in large models.

- If the model has 80 variables each with 10 levels, the joint distribution will have 10^{80} states $=$ the estimated number of atoms in the universe!
- In practice we a never interested in the joint distribution itself. We typically want the conditional distribution of one (or a few) variables given some of the other variables.
- We want to obtain this without calculating the joint distribution...

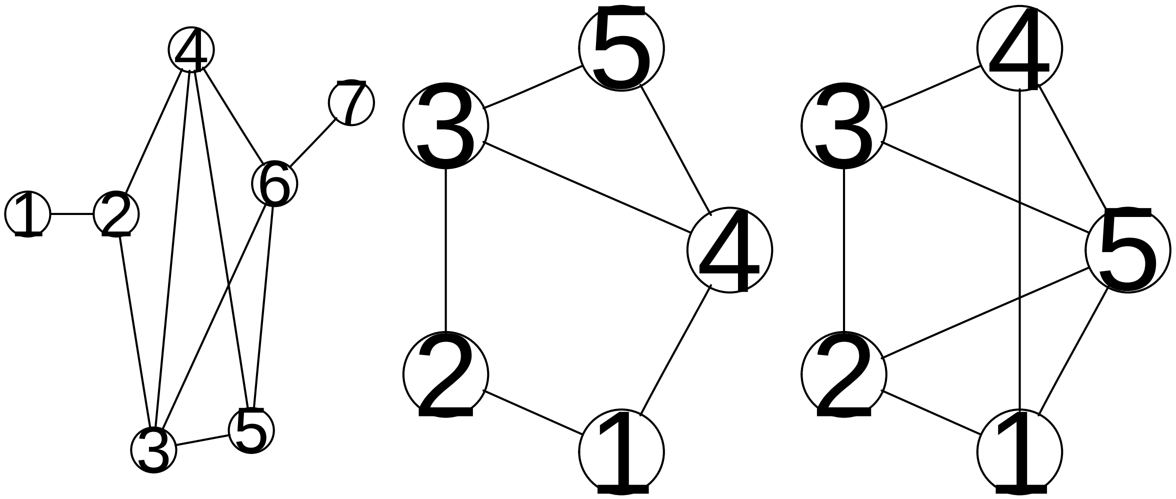
7 Decomposable graphs and junction trees

7.1 Decomposable graphs

Definition 11 *A graph is decomposable (or triangulated) if it contains no cycles of length* ≥ 4.

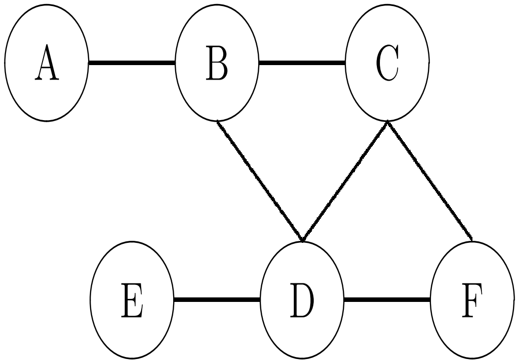
Decomposable graphs play a central role.

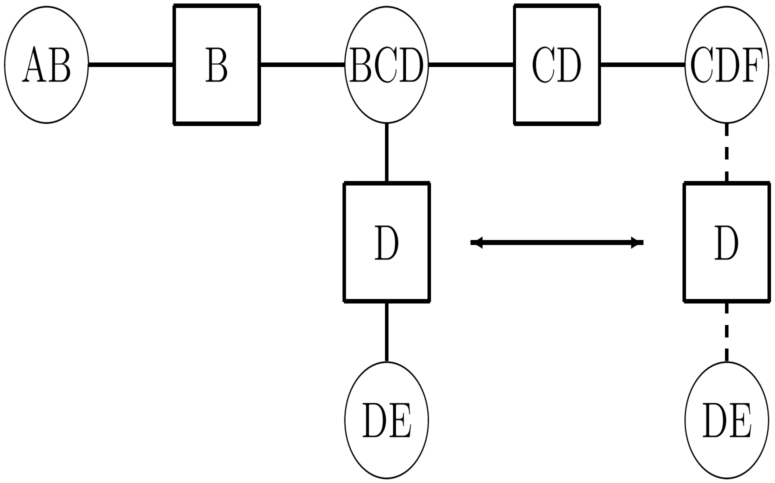
```
R> par(mfrow=c(1,3))
R> plot(ug(~1:2+2:3:4+3:4:5:6+6:7), "circo")  # decomposable
R> plot(ug(~1:2+2:3+3:4:5+4:1),"circo")       # not decomposable
R> plot(ug(~1:2:5+2:3:5+3:4:5+4:1:5),"circo") # not decomposable
```



7.2 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.

7.3 The key to message passing

Suppose

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

where C are the cliques of a decomposable graph (equivalently: nodes in the junction tree)

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 from DAG factorization:

- 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. $\psi_C(x_C, e^*) = p_C(x_C|e^*)$.

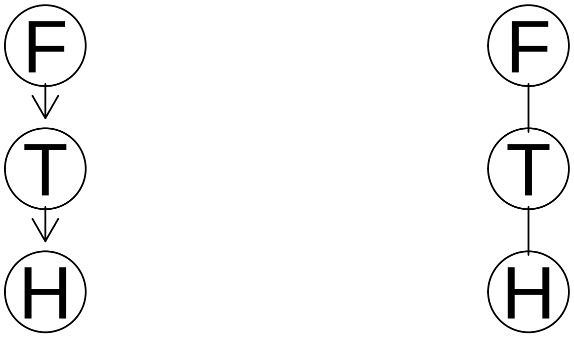
Done by *message passing* in *junction tree* .

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

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

7.4 Computations by message passing

```
R> par(mfrow=c(1,2), oma=c(2,1,2,1))
R> plot(FTH)
R> plot(moralize(FTH))
```



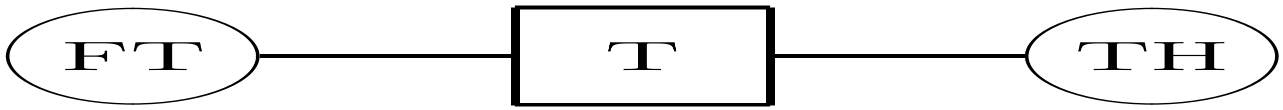
The moral graph is decomposable (essential for what follows).

Rewrite $p(V)$ as

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

where $\psi_T \equiv 1$.

Junction tree:



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

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

7.5 Clique potential representation

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

```
R> (qFT <- tableMult(p.F, p.TgF))

    F
T        F1      F2
  T1 0.0095 0.00099
  T2 0.0005 0.98901

R> (qTH <- p.HgT)

    T
H      T1   T2
  H1 0.8 0.01
  H2 0.2 0.99

R> (qT  <- parray("T",levels=2, values=1))

T
T1 T2
 1  1
```

7.6 Working inwards in junction tree

Work inwards towards root (i.e. from FT towards TH):

Set $\psi^*_T = \sum_F \psi_{FT}$.

Set $\psi^*_{TH} = \psi_{TH} \frac{\psi^*_T}{\psi_T}$

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 $p(T, H)$:

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

```
R> (qTs  <- tableMargin(qFT, "T"))

T
     T1      T2
0.01049 0.98951

R> (qTHs <- tableMult(qTH, tableDiv(qTs, qT)))

    H
T           H1        H2
  T1 0.0083920 0.0020980
  T2 0.0098951 0.9796149
```

7.7 Working outwards in junction tree

Work outwards from root (i.e. from TH towards FT):

Set $\psi_T^{**} = \sum_H \psi_{TH}^*$. Since $\psi_{TH}^* = p(T, H)$ we have

$$\psi_T^{**} = p(T) \quad \checkmark$$

```
R> (qTss <- tableMargin(qTHs, "T"))

T
     T1      T2
0.01049 0.98951
```

Set $\psi^*_{FT} = \psi_{FT}\frac{\psi^{**}_T}{\psi^*_T}$. Then

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

and

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

```
R> (qFTs <- tableMult(qFT, tableDiv(qTss, qTs)))

    F
T          F1      F2
  T1 0.0095 0.00099
  T2 0.0005 0.98901
```

This leaves us with marginal distributions on all cliques and separators

```
R> qFTs

    F
T        F1      F2
  T1 0.0095 0.00099
  T2 0.0005 0.98901

R> qTHs

    H
T           H1        H2
  T1 0.0083920 0.0020980
  T2 0.0098951 0.9796149

R> qTs

T
     T1      T2
0.01049 0.98951
```

From this we get:

```
R> qTs                   # probability of temperature

T
     T1      T2
0.01049 0.98951

R> tableMargin(qFT, "F") # probability of fever

F
  F1   F2
0.01 0.99

R> tableMargin(qTH, "H") # probability of headache

H
  H1   H2
0.81 1.19
```

– and we never calculated the joint distribution!!

8 Propagating findings

Suppose we have the finding $H = yes (= H1)$.

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

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

Repeat the computations above...

```
R> qTH

    T
H     T1   T2
  H1 0.8 0.01
  H2 0.2 0.99

R> ## Set finding H=H1
R> qTH[c(2,4)] <- 0
R> qTH

    T
H     T1   T2
  H1 0.8 0.01
  H2 0.0 0.00
```

```
R> ## Repeat everything
R> (qTs  <- tableMargin(qFT, "T"))

T
     T1      T2
0.01049 0.98951

R> (qTHs <- tableMult(qTH, tableDiv(qTs, qT)))

    H
T            H1 H2
  T1 0.0083920  0
  T2 0.0098951  0

R> (qTss <- tableMargin(qTHs, "T"))

T
       T1        T2
0.0083920 0.0098951

R> (qFTs <- tableMult(qFT, tableDiv(qTss, qTs)))

    F
T          F1        F2
  T1 7.6e-03 0.0007920
  T2 5.0e-06 0.0098901
```

After these operations, the tables only contain the clique probabilities up to a normalizing constant, i.e. $\psi_C(x_C) \propto p(x_C)$:

```
R> sum(qFTs)

[1] 0.0182871
```

To get probability of fever we must normalize:

```
R> tableMargin(qFTs, "F")/sum(qFTs)

F
       F1        F2
0.4158669 0.5841331
```

The important point of the computations: After working inwards and outwards in the junction tree, the clique potentials are consistent: They match on their separators:

```
R> tableMargin(qFTs, "T")

T
       T1        T2
0.0083920 0.0098951

R> tableMargin(qTHs, "T")

T
       T1        T2
0.0083920 0.0098951

R> qTss

T
       T1        T2
0.0083920 0.0098951
```

9 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 n*either* does the presence or absence of dyspnoea.”

A formalization of this narrative is as follows:

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

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

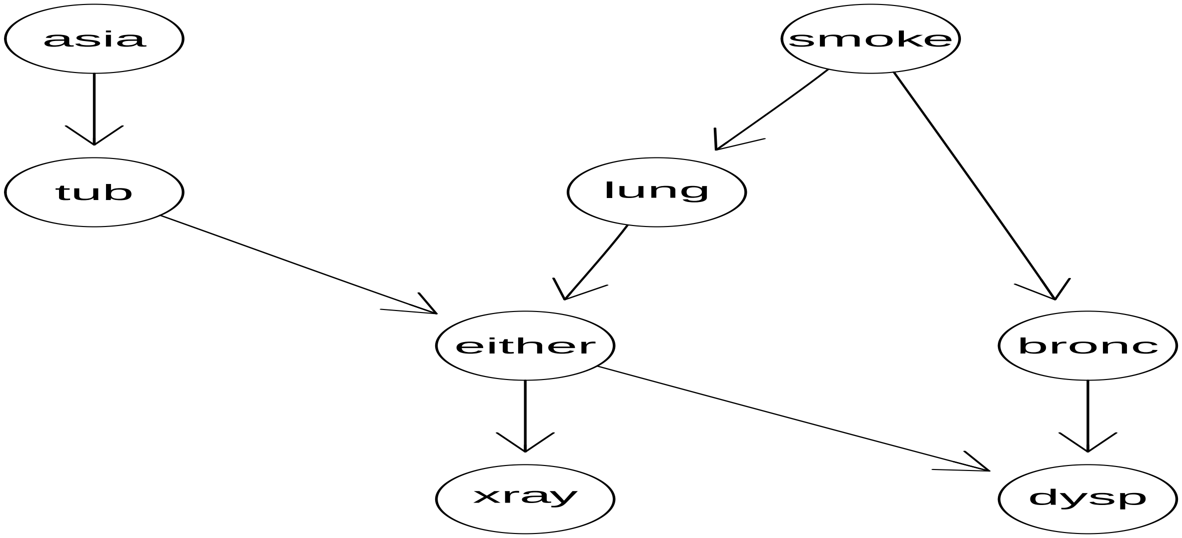


Figure 1: The directed acyclic graph corresponding to the chest clinic example.

9.1 Findings and queries

- Suppose we are given the *finding* that a person has recently visited Asia and suffers from dyspnoea, i.e. $A = \text{yes}$ and $D = \text{yes}$. Generally denote findings as $E = e^*$
- Interest may be in the conditional distributions $p(L \mid e^*)$, $p(T \mid e^*)$ and $p(B \mid e^*)$, or possibly in the joint (conditional) distribution $p(L, T, B \mid e^*)$.
- 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.

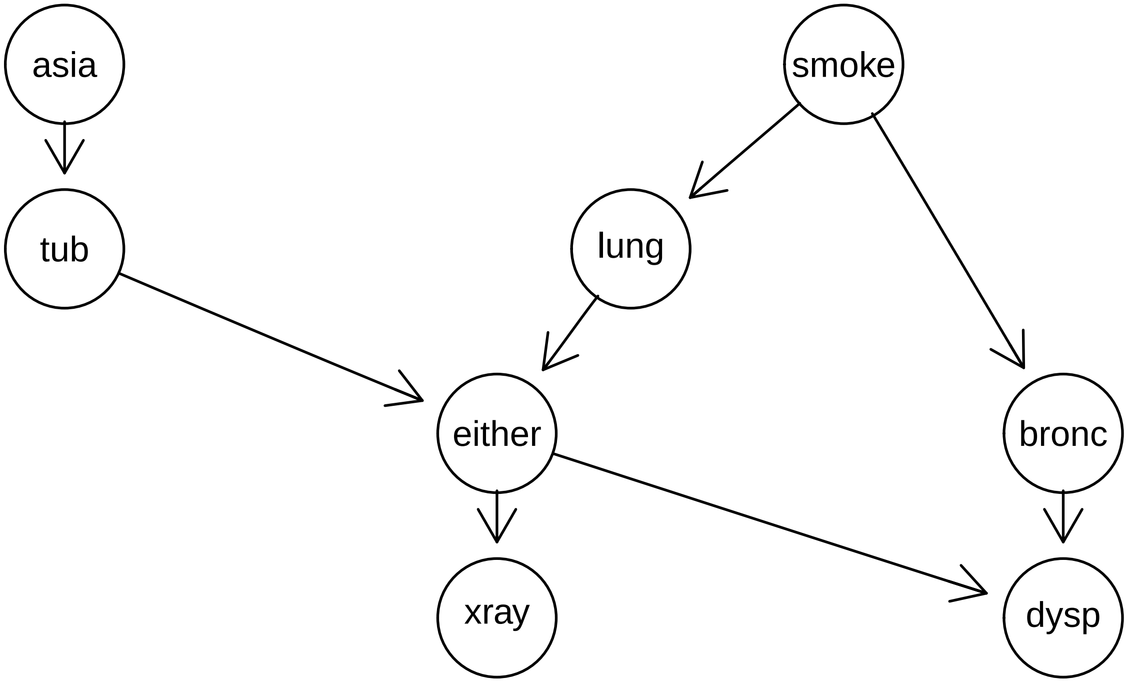
10 An introduction to the **gRain** package

Specify chest clinic network.

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

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

```
R> 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 )

R> plist$tub

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

```
R> plot(bnet)
```



10.1 Queries

```
R> 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
```

10.2 Setting findings and probability of findings

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

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

R> pFinding(bnet.f)

[1] 0.004501375
```

10.3 Queries – II

```
R> 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
```

```
R> 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
```

10.4 Dependence graph, moralization and triangulation

The computational scheme outlined above does not apply directly to the chest clinic example.

An extra step is needed: Triangulation of the moral graph.

Recall chest clinic model

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

Absorb lower order terms into higher order terms:

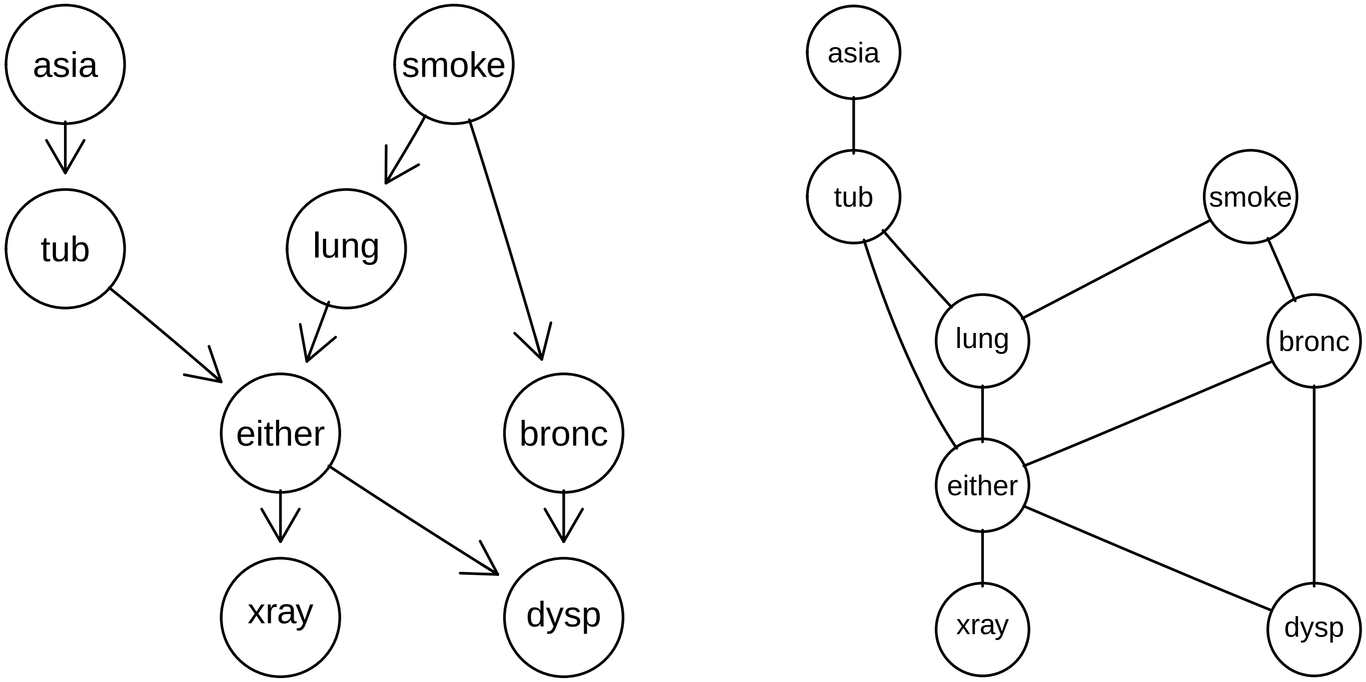
$$p(V) = \psi(T,A)\psi(L,S)\psi(B,S)\psi(E,T,L)\psi(D,E,B)\psi(X,E).$$

The dependence graph corresponding to the factorization

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

is the moral graph - but this graph is NOT triangulated:

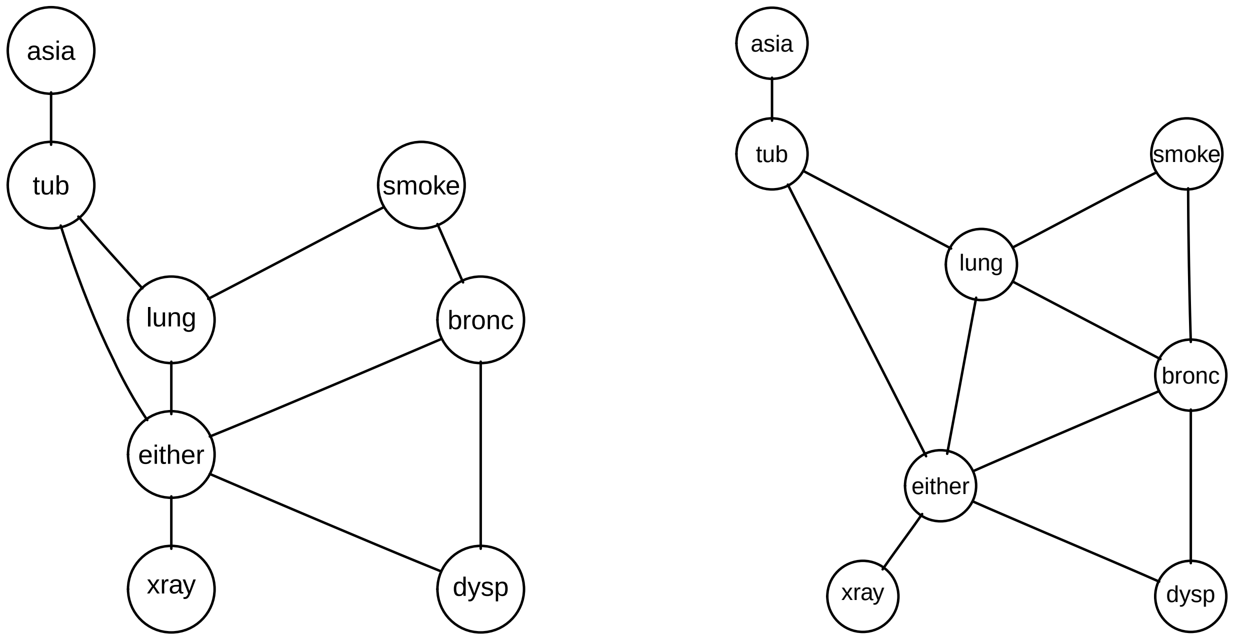
```
R> par(mfrow=c(1,2))
R> plot(bnet$dag)
R> plot(moralize(bnet$dag))
```



10.5 Triangulation

We can add edges, so called *fill–ins* , to the dependence graph to make the graph triangulated. This is called *triangulation* :

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

10.6 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.

11 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.

12 Contingency tables

In a study of lizard behaviour, characteristics of 409 lizards were recorded, namely species (S), perch diameter (D) and perch height (H). We have $V = \{D, H, S\}$.

```
R> data(lizardRAW, package="gRbase")
R> 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

R> dim(lizardRAW)

[1] 409   3
```

We may summarize data in a *contingency table* with cells (dhs) and counts n_{dhs} given by:

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

12.1 Notation

We consider a *discrete random vector* $X = (X_v; v \in V)$ where each X_v has a finite state space $\mathcal{X}_v$

A *configuration* of X is denoted by $x = (x_v, v \in V)$.

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.

The probability of an observation in cell x is denoted $p(x)$.

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

A *configuration* of X_A is denoted by x_A.

For $A \subset V$ we correspondingly have a *marginal table* with counts $n(x_A)$.

The probability of an observation in a marginal cell x_A is denoted $p(x_A) = \sum_{x' : x'_A = x_A} p(x')$.

```
R> 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

R> ## Marginal table
R> tableMargin(lizard, c("species","height"))

       height
species >4.75 <=4.75
  anoli    43    121
  dist    102    143
```

12.2 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

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

If no interaction terms are set to zero we have the *saturated model* .

If all interaction models are set to zero we have the *independence model*

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 $\mathcal{A} = \{A_1, \dots, A_Q\}$ 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

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

The MLE for $p(x)$ is the (unique) solution to the likelihood equations

$$\hat{p}(x_A) = n(x_A)/n, \quad A \in \mathcal{A}$$

Typically MLE must be found by iterative methods, e.g. iterative proportional scaling (IPS)

Iterative proportional scaling is implemented in *loglin()* :

```
R> (ll1 <- loglin(lizard, list(c("species","diam"),c("species","height"))))

2 iterations: deviation 0
$lrt
[1] 2.025647

$pearson
[1] 2.017364

$df
[1] 2

$margin
$margin[[1]]
[1] "species" "diam"

$margin[[2]]
[1] "species" "height"
```

A formula based interface to *loglin()* is provided by *loglm()* :

```
R> (ll2 <- loglm(~species:diam+species:height, data=lizard))

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


Statistics:
                      X^2 df  P(> X^2)
Likelihood Ratio 2.025647  2 0.3631921
Pearson          2.017364  2 0.3646994
```

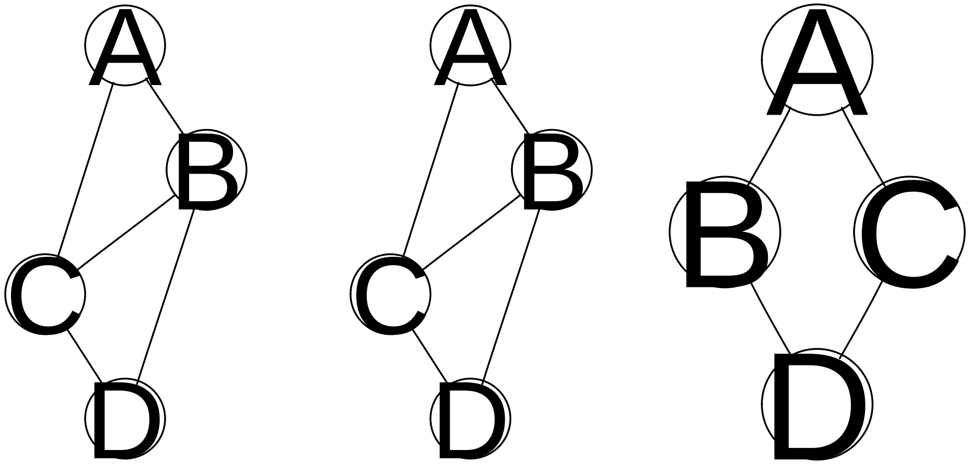
12.3 Graphical models and decomposable models

Definition 12 *A hierarchical log–linear model with generating class* $\mathcal{A} = \{a_1, \dots a_Q\}$ *is* graphical *if* $\mathcal{A}$ *are the cliques of the dependence graph.*

Definition 13 *A graphical log–linear model is* decomposable *if the models dependence graph is triangulated.*

Example 12.1 $\mathcal{A}_1 = \{ABC, BCD\}$ is graphical but $\mathcal{A}_2 = \{AB, AC, BCD\}$ is not. (Both have dependence graph with cliques $\mathcal{A}_1$). $\mathcal{A}_1$ is also decomposable. $\mathcal{A}_3 = \{AB, AC, BD, CD\}$ is graphical but not decomposable.

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



□

12.4 ML estimation in decomposable models

Consider model $\mathcal{A}_1 = \{ABC, BCD\}$. Index levels of A, B, C, D by i, j, k, l.

The MLE for this model is

$$\hat{p}_{ijkl} = \frac{\frac{n_{ijk+}}{n}\frac{n_{+jkl}}{n}}{\frac{n_{+jk+}}{n}}$$

- $\frac{n_{ijk+}}{n}$ is MLE $\hat{p}_{ijk}$ under the marginal model $\{ABC\}$ for ABC marginal table.
- $\frac{n_{+jkl}}{n}$ is MLE $\hat{p}_{jkl}$ under the marginal model $\{BCD\}$ for the BCD marginal table.
- $\frac{n_{+jk+}}{n}$ is MLE $\hat{p}_{jk}$ under the marginal model $\{BC\}$ for the BC marginal table.
- Generally, for a decomposable model, the MLE can be found in closed form as

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

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

Example 12.2 Consider the lizard data and the model $\mathcal{A} = \{[DS][HS]\}$. The MLE is

$$\hat{p}_{dhs} = \frac{(n_{d+s}/n)(n_{+hs}/n)}{n_{++s}/n} = \frac{n_{d+s}n_{+hs}}{nn_{++s}}$$

```
R> n.ds <- tableMargin(lizard, c("diam",   "species"))
R> n.hs <- tableMargin(lizard, c("height", "species"))
R> n.s  <- tableMargin(lizard, c("species"))
R> ## Expected cell counts
R> (fv   <- tableDiv( tableMult(n.ds, n.hs), n.s))

, , diam = <=4


       height
species    >4.75   <=4.75
  anoli 30.93902 87.06098
  dist  55.78776 78.21224


, , diam = >4


       height
species    >4.75   <=4.75
  anoli 12.06098 33.93902
  dist  46.21224 64.78776
```

```
R> as.data.frame.table(tablePerm(fv, c("diam","height","species")))

  diam height species     Freq
1  <=4  >4.75   anoli 30.93902
2   >4  >4.75   anoli 12.06098
3  <=4 <=4.75   anoli 87.06098
4   >4 <=4.75   anoli 33.93902
5  <=4  >4.75    dist 55.78776
6   >4  >4.75    dist 46.21224
7  <=4 <=4.75    dist 78.21224
8   >4 <=4.75    dist 64.78776

R> as.data.frame.table(fitted(l12))

Re-fitting to get fitted values
  diam height species     Freq
1  <=4  >4.75   anoli 30.93902
2   >4  >4.75   anoli 12.06098
3  <=4 <=4.75   anoli 87.06098
4   >4 <=4.75   anoli 33.93902
5  <=4  >4.75    dist 55.78776
6   >4  >4.75    dist 46.21224
7  <=4 <=4.75    dist 78.21224
8   >4 <=4.75    dist 64.78776
```

□

12.5 Connecting decomposable models and Bayesian networks

For a decomposable model, the MLE is given as

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

- The result (3) is IMPORTANT in connection with Bayesian networks, because (3) is a *clique potential* representation of p.
- Hence if we find a decomposable graphical model then we can convert this to a Bayesian network.
- We need not specify conditional probability tables (they are only used for specifying the model anyway, the real computations takes place in the junction tree).

13 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()).

```
R> 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.

```
R> 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:

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

13.1 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 $u \perp\!\!\!\perp v \mid S$ 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.

```
R> cit <- ciTest(lizard, set=~diam+height+species, slice.info=T)
R> cit

Testing diam _|_ height | species
Statistic (DEV):    2.026 df: 2 p-value: 0.3632 method: CHISQ

R> names(cit)

[1] "statistic" "p.value"   "df"        "statname"  "method"    "adjust.df"
[7] "varNames"  "slice"

R> 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 $\{n_{ijs}\}_{i=1\ldots|u|,j=1\ldots|v|}$. 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.

14 Log–linear models using the **gRim** package

```
R> data(wine, package="gRbase")
R> 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

R> 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:

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

R> dim(xtabs(~.,wine))

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

Just look at some variables

```
R> wine <- wine[,1:4]
R> 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

R> 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:

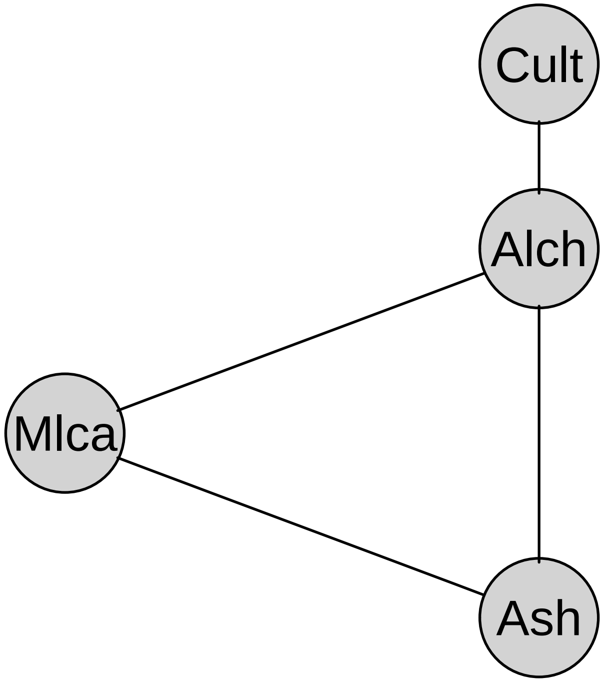
```
R> mm <- dmod(~Cult:Alch+Alch:Mlca:Ash, data=wine)
R> mm <- dmod(list(c("Cult","Alch"), c("Alch","Mlca","Ash")), data=wine)
R> mm <- dmod(~C:Alc+Alc:M:As, data=wine)
R> mm

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()` :

```
R> str(terms(mm))

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

R> formula(mm)

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

14.1 Plotting the dependence graph

```
R> plot(mm)
```



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

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

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

R> # Alternative: an adjacency matrix
R> 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
```

14.2 Model specification shortcuts

Shortcuts for specifying some models

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

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

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

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

R> 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"
```

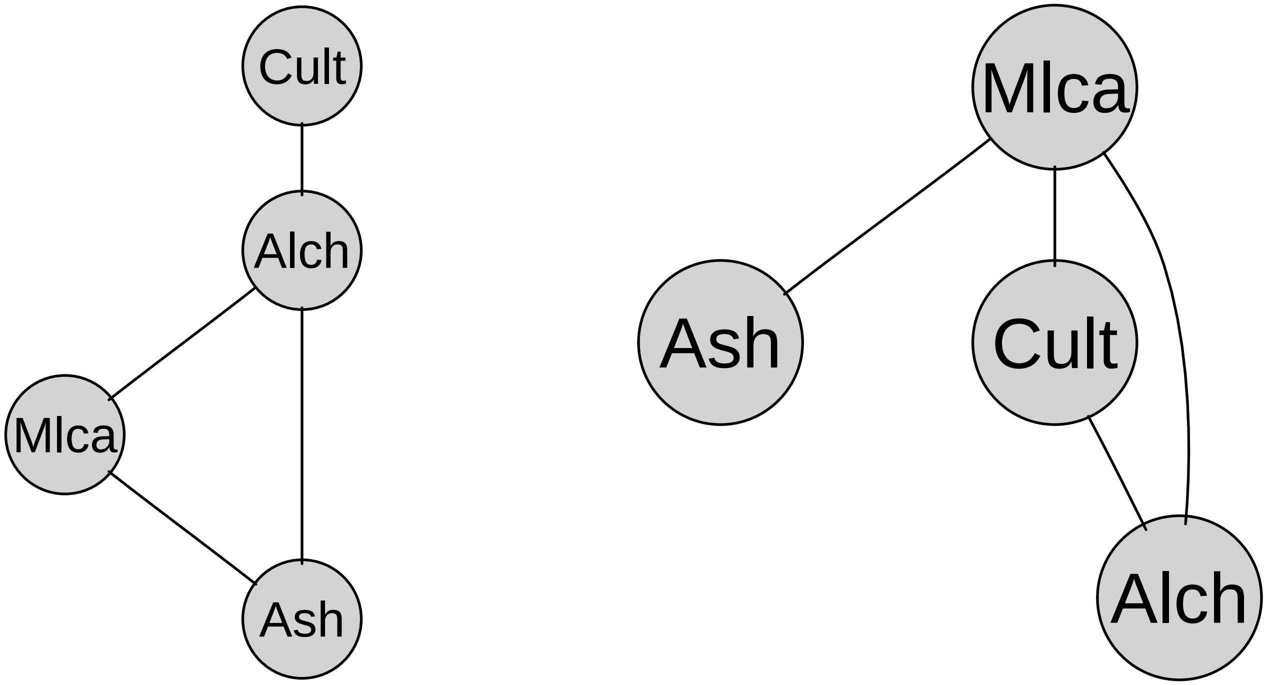
Useful to combine with specification of a marginal table:

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

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

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

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

14.3 Altering graphical models

Natural operations on graphical models: add and delete edges

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



14.4 Model comparison

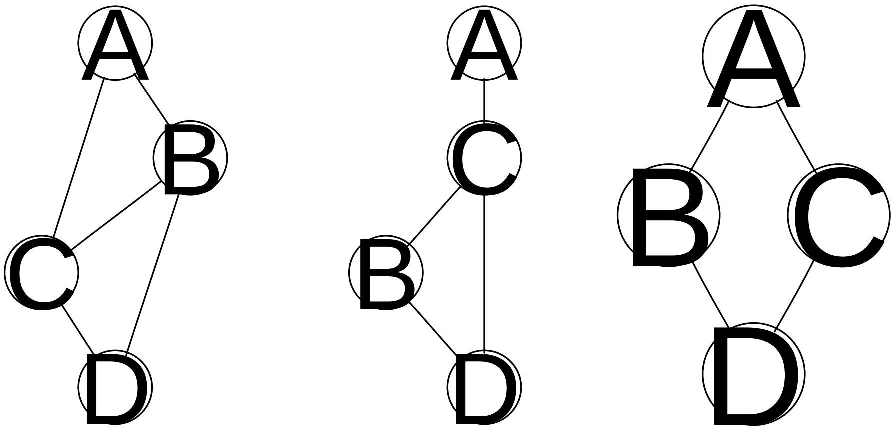
Models are compared with compareModels() .

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

14.5 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.

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

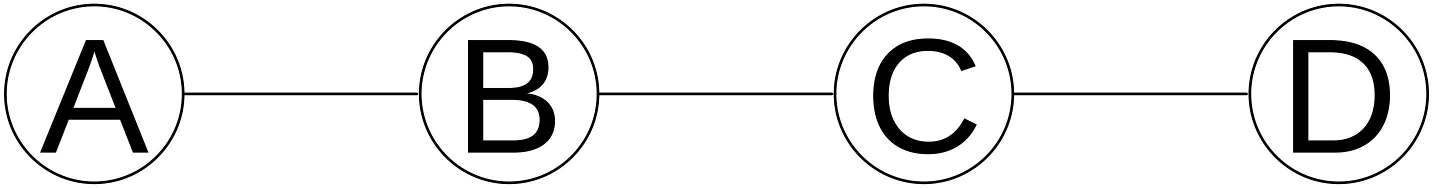
Result: The test for removal of $e = \{u, v\}$ which is contained in one clique C only can be made as a test for $u \perp\!\!\!\perp v | C \setminus \{u, v\}$ in the C–marginal table.

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

14.6 Decomposable models – adding edges

More tricky when adding edge to a decomposable model

```
R> plot(ug(~A:B+B:C+C:D), "circo")
```



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.

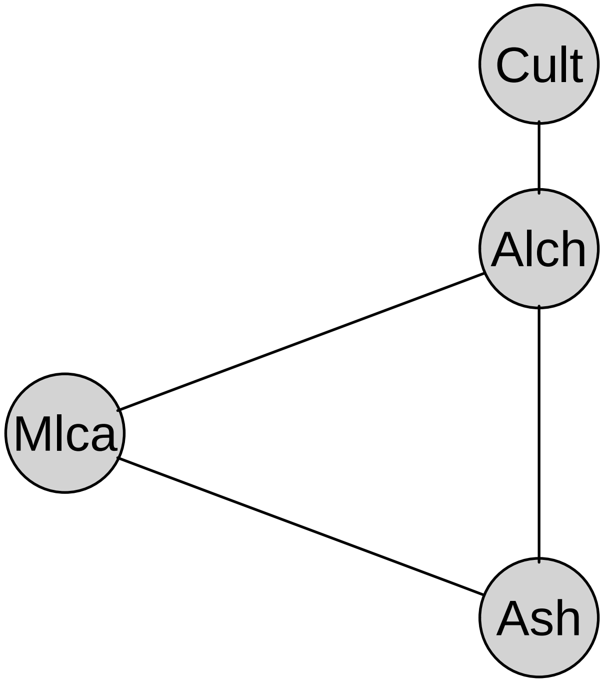
Can be tested with *maximum cardinality search* as implemented in `mcs()` . Runs in $O(|edges| + |vertices|)$.

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

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

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

character(0)

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

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

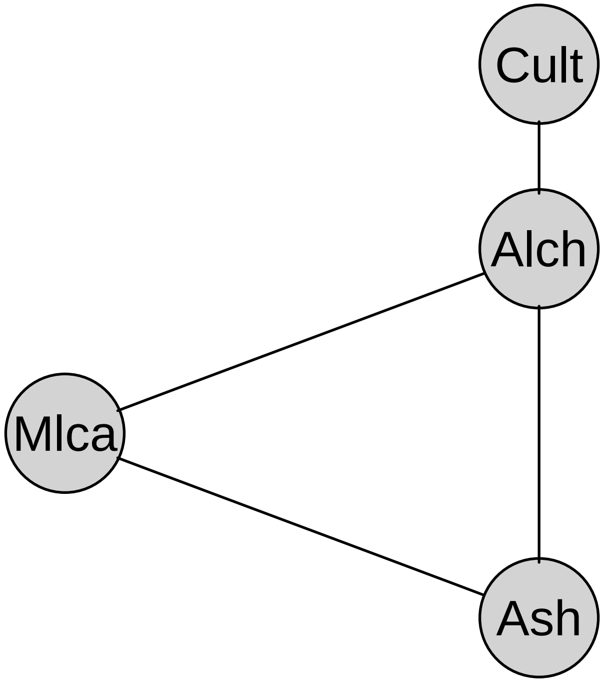
14.7 Test for adding and deleting edges

Done with `testdelete()` and `testadd()`

```
R> mm <- dmod(~C:Alc+Alc:M:As, data=wine)
R> plot(mm)
R> 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
```



```
R> mm <- dmod(~C:Alc+Alc:M:As, data=wine)
R> plot(mm)
R> 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
```



14.8 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))

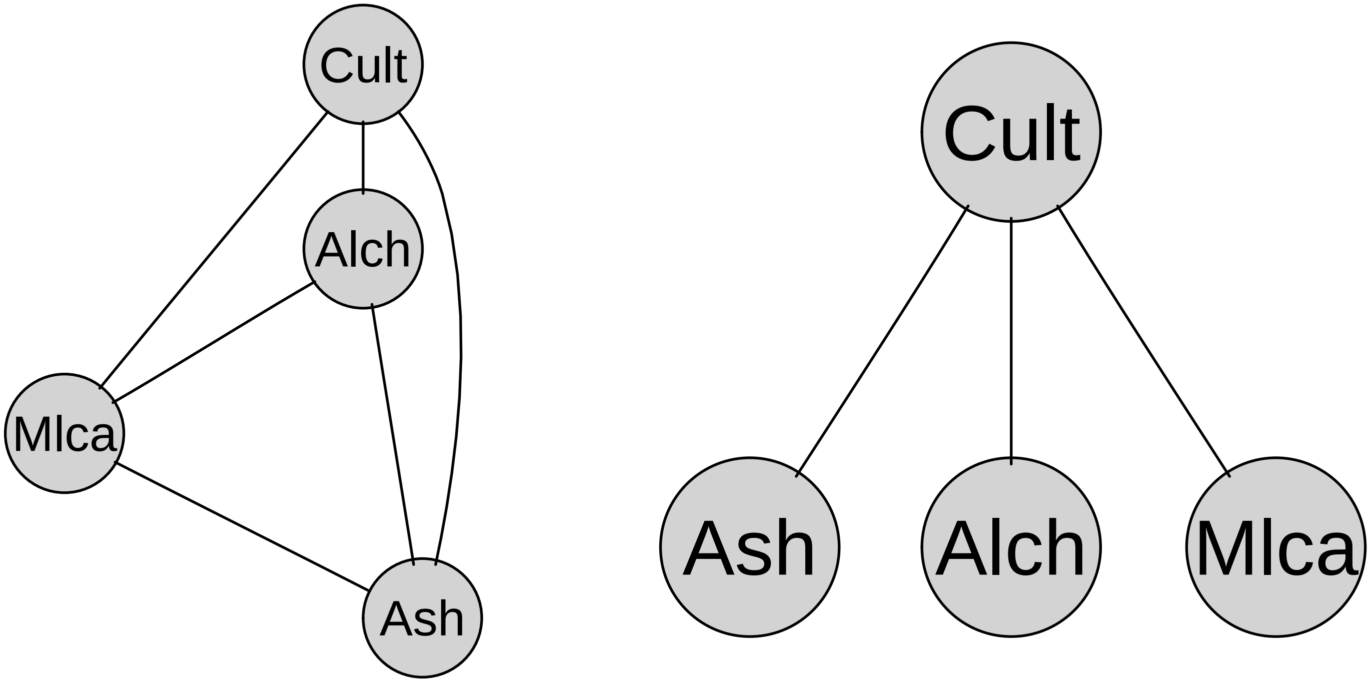
- Model types can be ”unsrestricted” or ”decomposable”. (Default is decomposable if initial model is decompsable)

- Search method can be ”all” or ”headlong”. (Default is all)

```
R> 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
```

```
R> dm1 <- dmod(~.^., data=wine)
R> 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

R> formula(dm2)

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

R> terms(dm2)

[[1]]
[1] "Cult" "Ash"


[[2]]
[1] "Cult" "Alch"


[[3]]
[1] "Cult" "Mlca"
```

```
R> par(mfrow=c(1,2))
R> plot(dm1, "circo")
R> plot(dm2, "circo")
```



15 From graph and data to network

Create graphs from models:

```
R> uG2 <- ugList(terms(dm2))
R> uG2 <- ugList(list(c("Cult", "Ash"), c("Cult", "Alch"), c("Cult", "Mlca")))
R> uG1 <- ugList(terms(dm1))
```

Given a graph (either an undirected decomposable graph or a DAG) and data we can construct BN's on the fly (data can be a dataframe or a table)

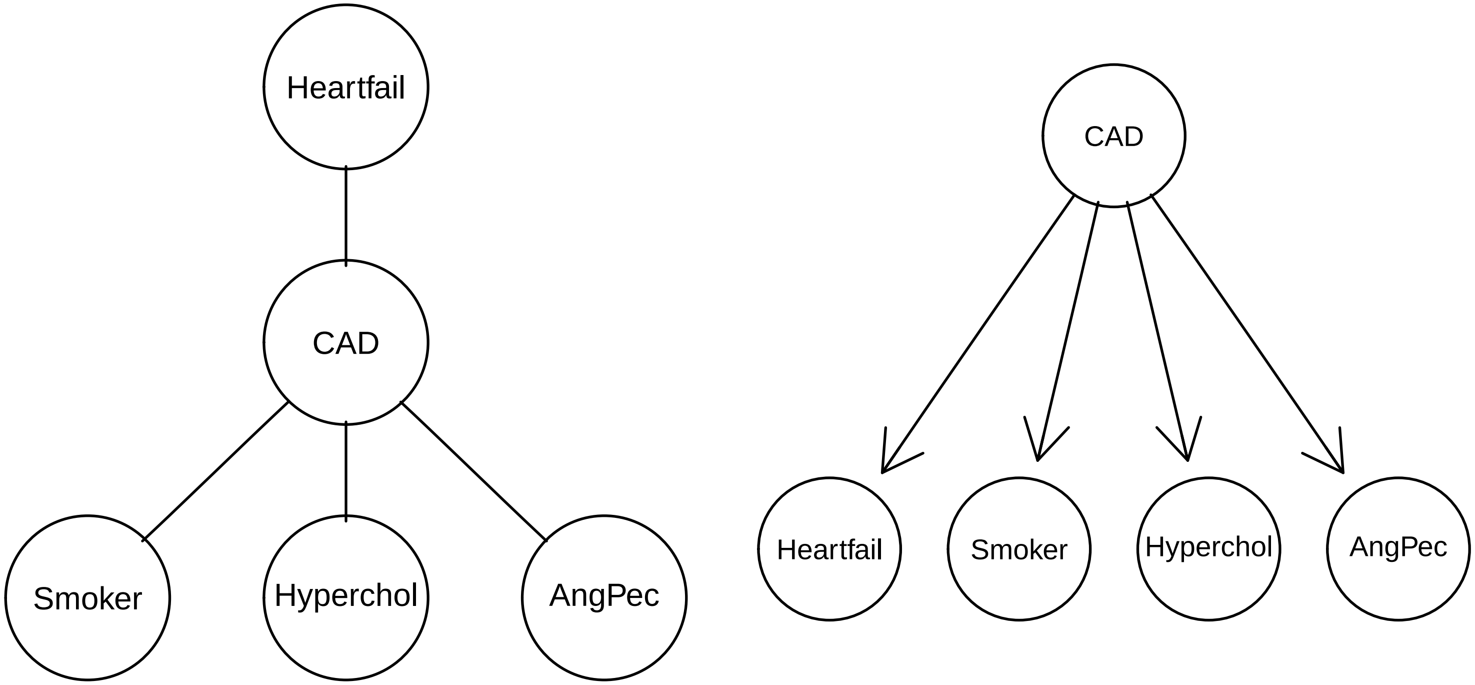
```
R> (wine1 <- compile(grain(uG1, data=wine)))

Independence network: Compiled: TRUE Propagated: FALSE
 Nodes: chr [1:4] "Ash" "Cult" "Alch" "Mlca"

R> (wine2 <- compile(grain(uG2, data=wine)))

Independence network: Compiled: TRUE Propagated: FALSE
 Nodes: chr [1:4] "Ash" "Cult" "Alch" "Mlca"
```

```
R> querygrain(wine1, "Cult")

$Cult
Cult
       v1        v2        v3
0.3314607 0.3988764 0.2696629

R> querygrain(wine2, "Cult")

$Cult
Cult
       v1        v2        v3
0.3314607 0.3988764 0.2696629
```

```
R> querygrain(setFinding(wine1, c("Ash","Alch"), c("L","H")), "Cult")

$Cult
Cult
       v1        v2        v3
0.5769231 0.1923077 0.2307692

R> querygrain(setFinding(wine2, c("Ash","Alch"), c("L","H")), "Cult")

$Cult
Cult
       v1        v2        v3
0.5524341 0.2029550 0.2446109
```

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

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

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

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

$CAD
CAD
       No       Yes
0.5466102 0.4533898

R> querygrain(cadmod2, nodes="CAD")

$CAD
CAD
       No       Yes
0.5466102 0.4533898
```

15.1 Prediction

We shall try to predict CAD in the validation dataset cad2

```
R> data(cad2)
R> 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() .

```
R> args(predict.grain)

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

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

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

15.2 Classification error

```
R> table(cad2$CAD)

 No Yes
 41  26

R> table(cad2$CAD)/sum(table(cad2$CAD))

       No       Yes
0.6119403 0.3880597

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

      No Yes
  No  34   7
  Yes 14  12

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

             No       Yes
  No  0.8292683 0.1707317
  Yes 0.5384615 0.4615385
```

16 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 **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.

17 Practicals

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

```
R> data(cad1)
R> head(cad1,5)

     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
  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
```

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

```
R> data(cad2)
R> head(cad2,5)

     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
  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
```

- Start out from the saturated model; do stepwise backward model selection among decomposable models to obtain “reduced” model.

- Start out from the independence model; do stepwise forward model selection among decomposable models to obtain “expanded” model.

- Can you identify direct and indirect risk factors from these models?

- “Convert” these two models to a Bayesian nework.

- Predict the disease variable `CAD` in the validation dataset `cad2` (which has incomplete observations).

- How good prediction results can you obtain? Can you improve your result by changing the model selection criterion, for example going from AIC to BIC?

- Create a “naive Bayes model”: All predictor variables are independent given `CAD`. How does this model perform in predicting the validation cases?

- For the curious: Create a recursive partitioning tree using **rpart** (see supplementary notes).
- Does such trees perform better than the graphical models?
- Discuss pros and cons of the approaches