Graphical Models with R 1st talk: Graphs and Markov properties with R

Presentation · July 2018

1 author:

Dhafer Malouche
University of Carthage

Graphical Models with R

1st talk: Graphs and Markov properties with R

Dhafer Malouche

essai.academia.edu/DhaferMalouche

Table of contents

Terminology of graphs

What's a graph?

- A graph is $\mathcal{G} = (V, E)$ where V is a finite set and $E \subseteq V \times V$.
  - V the of vertices
  - E is the set edges (its elements are denoted by $\alpha\beta$)
- undirected when $\alpha - \beta$:

$$\alpha\beta \in E \iff \beta\alpha \in E$$

- directed when $\alpha \to \beta$ or $\alpha \leftarrow \beta$

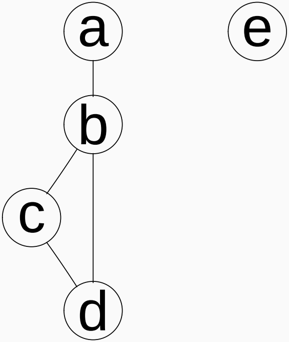
$$\text{If } \alpha\beta \in E \Rightarrow \beta\alpha \notin E$$

- bi-directed when $\alpha \leftrightarrow \beta$

Undirected Graphs

Undirected Graph (UG)

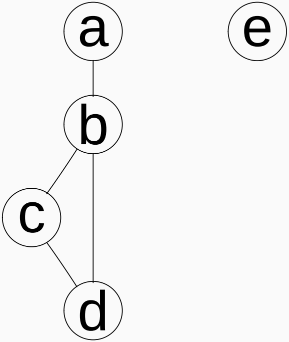
- $V = \{a, b, c, d, e\}$
- $E = \{ab, bc, cd, bd\}$
- cliques: ab, bcd, e



Undirected Graph (UG)

- $V = \{a, b, c, d, e\}$
- $E = \{ab, bc, cd, bd\}$
- cliques: *ab*, *bcd*, *e*

A *clique* in $\mathcal{G}$ is a maximal complete subset of V



UG with R, `gRbase`

- `graphNEL` objects
- `ug()` function

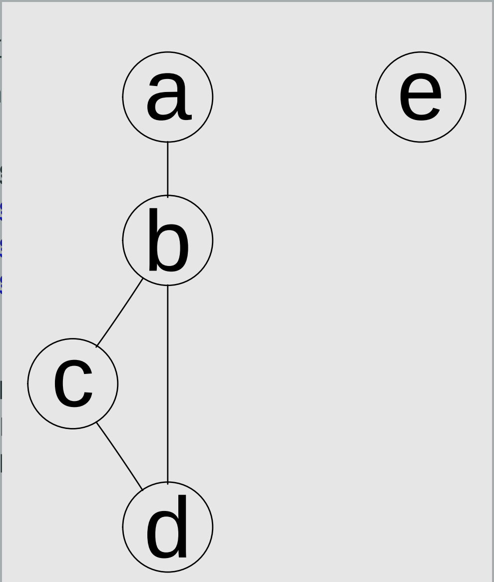
UG with R, gRbase

- `graphNEL` objects
- `ug()` function

```
> library(gRbase)
> ug0 <- ug(~a:b,~b:c:d,~e)
> ug0 <- ug(~a:b+b:c:d+e)
> ug0 <- ug(c("a","b"),c("b","c","d"),"e")
> ug0
A graphNEL graph with undirected edges
Number of Nodes = 5
Number of Edges = 4
> library(Rgraphviz)
> plot(ug0)
```

- graphNE
- ug() fun

```
> library(
> ug0 <- u
> ug0 <- u
> ug0 <- u                                    )
> ug0
A graphNEL
Number of
Number of
> library(
> plot(ug0
```



Adjacency matrix with R, gRbase

$$\mathcal{G} = (V, E) \longmapsto A = [a(\alpha\beta)] \in \{0,1\}^{|V|\times|V|} \quad \text{such that}$$

$$a(\alpha\beta) = 1 \iff \alpha\beta \in E$$

Adjacency matrix with R, gRbase

$$\mathcal{G} = (V, E) \longmapsto A = [a(\alpha\beta)] \in \{0,1\}^{|V|\times|V|} \quad \text{such that}$$

$$a(\alpha\beta) = 1 \iff \alpha\beta \in E$$

```
> ug01 <- ug(~a:b+b:c:d+e,result="matrix")
> ug01
  a b c d e
a 0 1 0 0 0
b 1 0 1 1 0
c 0 1 0 1 0
d 0 1 1 0 0
e 0 0 0 0 0
```

Nodes and Edges R, gRbase

```
> nodes(ug0)
[1] "a" "b" "c" "d" "e"
> edges(ug0)
$a
[1] "b"
$b
[1] "c" "d" "a"
$c
[1] "d" "b"
$d
[1] "b" "c"
$e
character(0)
```

UG with R, igraph

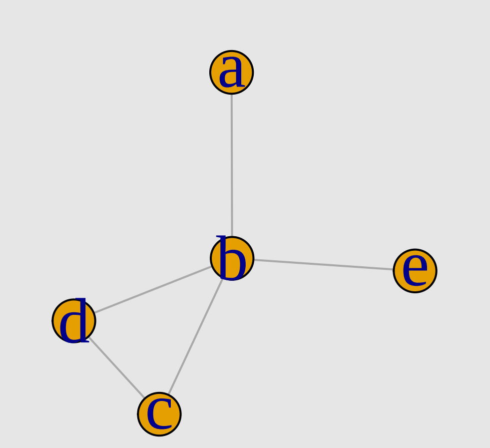
```
> ug0i <- ug(c("a","b"),c("b","c","d"),c("e","b"),
+               result="igraph")
> ug0i
IGRAPH UNW- 5 5 --
+ attr: name (v/c), label (v/c), weight (e/n)
+ edges (vertex names):
[1] a--b b--c b--d b--e c--d
> library(igraph)
> ## vertices
> V(ug0i)
+ 5/5 vertices, named:
[1] a b c d e
> ## edges
> E(ug0i)
+ 5/5 edges (vertex names):
[1] a--b b--c b--d b--e c--d
```

UG with R, igraph

```
> V(ug0i)$size <- 25
> V(ug0i)$label.cex <- 2
> plot(ug0i, layout=layout.spring)
```

UG with R, igraph

```
> V(ug0i)$s
> V(ug0i)$l
> plot(ug0i
```



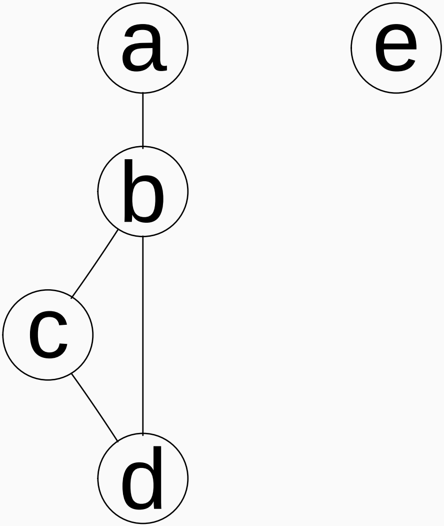
Cliques in an UG with R,

```
> library(RBGL)
> is.complete(ug0, c("b","c","d"))
[1] TRUE
> maxClique(ug0)
$maxCliques
$maxCliques[[1]]
[1] "b" "c" "d"

$maxCliques[[2]]
[1] "b" "a"

$maxCliques[[3]]
[1] "e"
```

Paths and separators in an UG

- 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 $\alpha_{i-1} \sim \alpha_i$ for $i = 1, \ldots, n$.
- If $\alpha = \beta$ then the path is said to be a cycle of length n.
- A subset $S \subset V$ in an undirected graph is said to separate $A \subseteq V$ from $B \subseteq V$ if every path between a vertex in A and a vertex in B intersects S.

Paths and separators in an UG



Paths and separators in an UG with R

```
> separates(a = "a", b = "d",S1 =  c("b", "c"), ug0)
[1] TRUE
> separates(a = "a", b = "b",S1 =  c("d", "c"), ug0)
[1] FALSE
```

Subgraphs

The graph $G_0 = (V_0, E_0)$ is said to be a subgraph of $G = (V, E)$ if $V_0 \subseteq V$ and $E_0 \subseteq E$.

Subgraphs

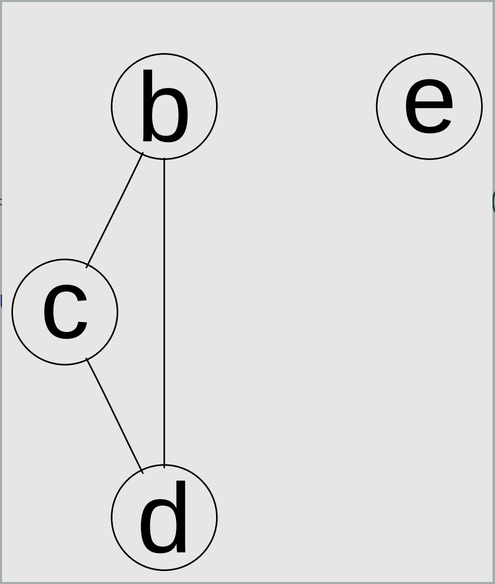
The graph $G_0 = (V_0, E_0)$ is said to be a subgraph of $G = (V, E)$ if $V_0 \subseteq V$ and $E_0 \subseteq E$.

```
> ug1 <- subGraph(c("b","c","d","e"), ug0)
> plot(ug1)
```

Subgraphs

The graph G_0 = … (V, E) if $V_0 \subseteq V$ and $E_0 \subseteq E$.

```
> ug1 <- su… )
> plot(ug1
```



Boundary

Boundary	$\mathrm{bd}(\alpha) = \{\beta \in V,\ \beta \sim \alpha\}$
Closure	$\mathrm{cl}(\alpha) = \mathrm{bd}(\alpha) \cup \{\alpha\}$

Boundary

Boundary $\quad \text{bd}(\alpha) = \{\beta \in V,\ \beta \sim \alpha\}$

Closure $\quad \text{cl}(\alpha) = \text{bd}(\alpha) \cup \{\alpha\}$

```
> adj(object = ug0, "c")
$c
[1] "d" "b"
> closure(object = ug0,set = "c")
[1] "c" "d" "b"
```

Simplicial, Connected components

A node in an undirected graph is simplicial if its boundary is complete.

Simplicial, Connected components

A node in an undirected graph is simplicial if its boundary is complete.

```
> is.simplicial(set = "b", object = ug0)
[1] FALSE
> simplicialNodes(object = ug0)
[1] "a" "c" "d" "e"
> connectedComp(g = ug0)
$`1`
[1] "a" "b" "c" "d"

$`2`
[1] "e"
```

A chord, a triangulated UG

- An edge $\alpha_i \sim \alpha_j$ is a chord if the nodes of this edge belong to a cycle $\alpha = \alpha_0 \sim \alpha_1 \sim \dots \alpha_n = \alpha$ and where $j \notin \{i-1, i+1\}$
- A graph where all the cycle of length ≥ 4 are *chorldless* is called triangulated graph

A chord, a triangulated UG

- An edge $\alpha_i \sim \alpha_j$ is a chord if the nodes of this edge belong to a cycle $\alpha = \alpha_0 \sim \alpha_1 \sim \ldots \alpha_n = \alpha$ and where $j \notin \{i-1, i+1\}$
- A graph where all the cycle of length ≥ 4 are *chorldless* is called triangulated graph

```
> is.triangulated(ug0)
[1] TRUE
```

Decomposition of an UG

Let (A, B, S) be a triplet of subsets of V. (A, B, S) is a decomposition of $\mathcal{G}$ if

i. (A, B, S) are disjoints and $V = A \cup B \cup S$
ii. S is complete
iii. S separates A and B in $\mathcal{G}$

Decomposition of an UG

Let (A, B, S) be a triplet of subsets of V. (A, B, S) is a decomposition of $\mathcal{G}$ if

i. (A, B, S) are disjoints and $V = A \cup B \cup S$
ii. S is complete
iii. S separates A and B in $\mathcal{G}$

```
> is.decomposition(set = "a", set2  = "d", set3 = c("b","c"), ug0)
[1] FALSE
> ug1<-subGraph(c("b","c","d","a"), ug0)
> is.decomposition(set = "a", set2  = "d", set3 = c("b","c"), ug1)
[1] TRUE
> is.decomposition(set = "a", set2  = c("d","b"), set3 = "c", ug1)
[1] FALSE
```

A decomposable UG

$\mathcal{G} = (V, E)$ is called a decomposable if

i. $\mathcal{G}$ is *complete*, i.e; $E = V \times V$.
ii. or it can be decomposed into a decomposable subgraphs.

A decomposable UG

$\mathcal{G} = (V, E)$ is called a decomposable if

i. $\mathcal{G}$ is *complete*, i.e; $E = V \times V$.

ii. or it can be decomposed into a decomposable subgraphs.

Theorem

$\mathcal{G}$ is decomposable if and only if $\mathcal{G}$ is triangulated

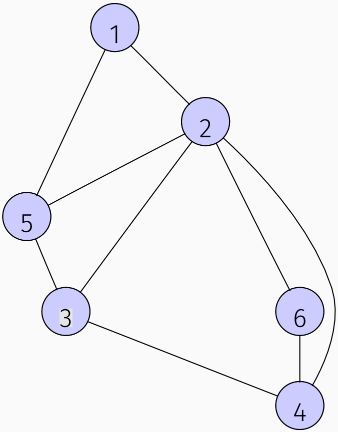
Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \mathrm{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \mathrm{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

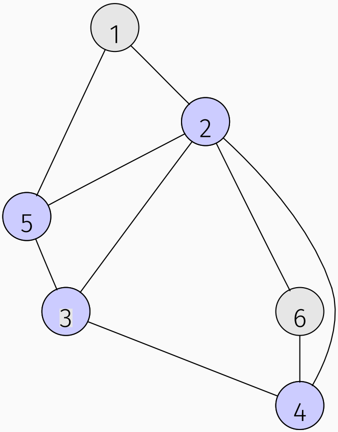
i	bd(i)	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y



Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \text{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

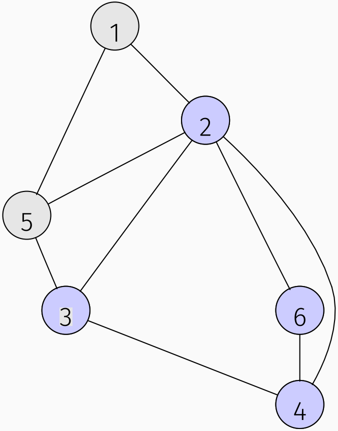
i	bd(i)	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y
3	$\{2, 3, 4, 5\}$	$\{2\}$	Y



Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \text{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

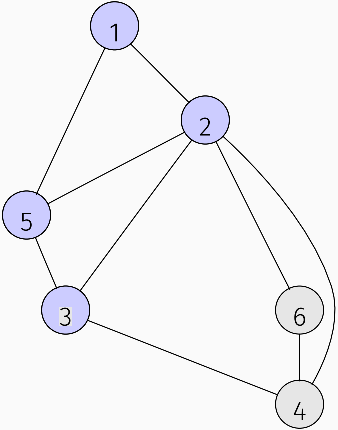
i	bd(i)	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y
3	$\{2, 3, 4, 5\}$	$\{2\}$	Y
4	$\{2, 3, 6\}$	$\{2, 3\}$	Y



Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \mathrm{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

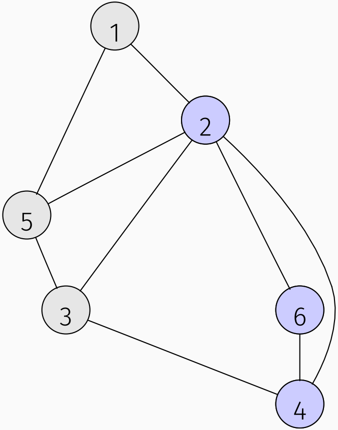
i	bd(i)	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y
3	$\{2, 3, 4, 5\}$	$\{2\}$	Y
4	$\{2, 3, 6\}$	$\{2, 3\}$	Y
5	$\{1, 2, 3\}$	$\{1, 2, 3\}$	N



Perfect ordering

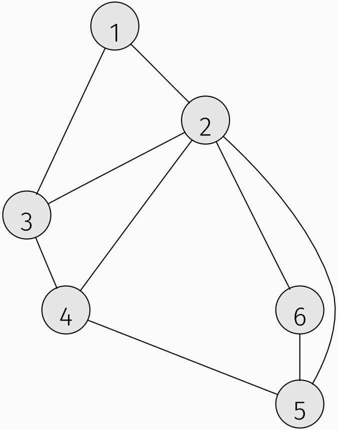
Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \text{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

i	$\text{bd}(i)$	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y
3	$\{2, 3, 4, 5\}$	$\{2\}$	Y
4	$\{2, 3, 6\}$	$\{2, 3\}$	Y
5	$\{1, 2, 3\}$	$\{1, 2, 3\}$	N
6	$\{2, 4\}$	$\{2, 4\}$	Y



Perfect ordering

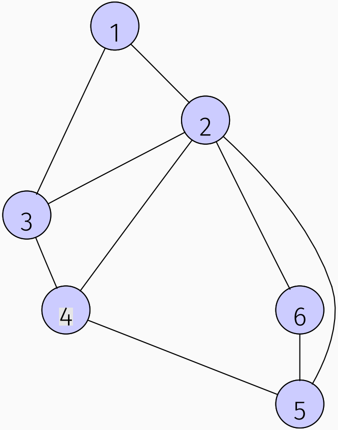
Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \mathrm{bd}(i) \cap \{1, \ldots, i-1\}$ is complete



Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \text{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

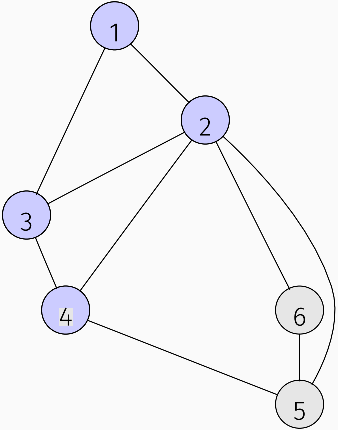
i	bd(i)	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y



Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \mathrm{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

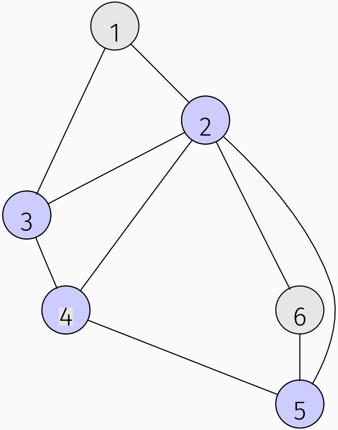
i	$\mathrm{bd}(i)$	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y
3	$\{1, 2, 4\}$	$\{1, 2\}$	Y



Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \mathrm{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

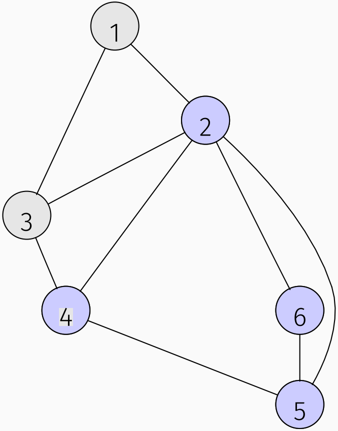
i	$\mathrm{bd}(i)$	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y
3	$\{1, 2, 4\}$	$\{1, 2\}$	Y
4	$\{2, 3, 5\}$	$\{2, 3\}$	Y



Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \text{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

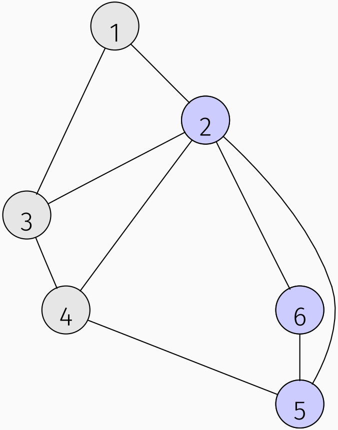
i	bd(i)	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y
3	$\{1, 2, 4\}$	$\{1, 2\}$	Y
4	$\{2, 3, 5\}$	$\{2, 3\}$	Y
5	$\{2, 4, 6\}$	$\{2\}$	Y



Perfect ordering

Assume $V = \{1, \ldots, |V|\}$. This is order called perfect if $\forall i = 2, \ldots, |V|$, $S(i) = \text{bd}(i) \cap \{1, \ldots, i-1\}$ is complete

i	bd(i)	$S(i)$	Complete?
2	$\{1, 3, 4, 5, 6\}$	$\{1\}$	Y
3	$\{1, 2, 4\}$	$\{1, 2\}$	Y
4	$\{2, 3, 5\}$	$\{2, 3\}$	Y
5	$\{2, 4, 6\}$	$\{2\}$	Y
6	$\{2, 5\}$	$\{2, 5\}$	Y



Perfect ordering and decomposability

- If $\mathcal{G}$ is decomposable, then the perfect ordering can be obtained using the *maximum cardinality search* algorithm

Perfect ordering and decomposability

- If $\mathcal{G}$ is decomposable, then the perfect ordering can be obtained using the *maximum cardinality search* algorithm

Theorem

$\mathcal{G}$ is decomposable if and only if $\mathcal{G}$ is triangulated if and only if the vertices of $\mathcal{G}$ admit a perfect ordering.

Perfect ordering and decomposability

- If $\mathcal{G}$ is decomposable, then the perfect ordering can be obtained using the *maximum cardinality search* algorithm

Theorem

$\mathcal{G}$ is decomposable if and only if $\mathcal{G}$ is triangulated if and only if the vertices of $\mathcal{G}$ admit a perfect ordering.

```
> g2<-ug(~1*2*5,~2*5*3,~2*6*4,~2*4*3)
> mcs(g2)
[1] "1" "2" "5" "3" "4" "6"
```

RIP ordering for the cliques

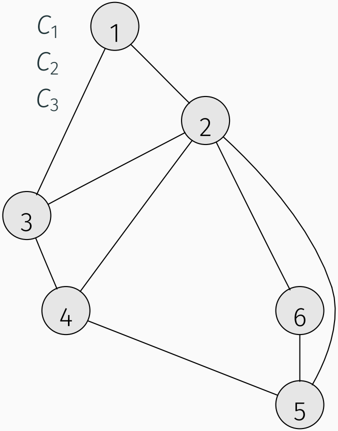
- Let $\{C_1, \ldots, C_p\}$ be the set of cliques for $\mathcal{G}$
- A Running Intersection Property of $\{C_1, \ldots, C_p\}$ means that for all $j = 2, \ldots, p$, $\exists\, i < j$ such that

$$C_j \cap \left(C_1 \cup \ldots \cup C_{j-1}\right) \subset C_i$$

- $S_1 = \emptyset$, $S_2 = C_2 \cap C_1$, $S_3 = C_3 \cap (C_1 \cup C_2)$, …, $S_p = C_p \cap (C_1 \cup \ldots \cup C_{p-1})$
- $R_1 = C_1$, $R_2 = C_2 \setminus S_2$, …,$R_j = C_j \setminus S_j$, …, $R_p = C_p \setminus S_p$.
- $S_2 = C_2 \cap C_1$ separates R_2 from $H_2 = C_1 \setminus S2$
- $\forall j \geq 2$, S_j separates R_j from $H_j = (C_1 \cup \ldots \cup C_{j-1}) \setminus S_j$.

RIP ordering for the cliques

j	C_j	S_j	R_j	H_j	Sep.	$\subset C_i$?
1	$\{1, 2, 3\}$	$\emptyset$				
2	$\{2, 3, 4\}$	$\{2, 3\}$	$\{4\}$	$\{1\}$	Y	C_1
3	$\{2, 4, 5\}$	$\{2, 4\}$	$\{5\}$	$\{1, 3\}$	Y	C_2
4	$\{2, 5, 6\}$	$\{2, 5\}$	$\{6\}$	$\{1, 3, 4\}$	Y	C_3



RIP ordering for the cliques

- If $\mathcal{G}$ is triangulated RIP ordering exists (iff)
- $\exists i < j$ such $S_j \subset C_i$, C_i is called the parent and the S_j are called separators.

```
> g1<-ug(~1*2*3,~2*3*4,~2*4*5,~2*6*5)
> rip(g1)
cliques
  1 : 2 3 1
  2 : 2 3 4
  3 : 2 5 4
  4 : 2 5 6
separators
  1 :
  2 : 2 3
  3 : 2 4
  4 : 2 5
parents
  1 : 0
  2 : 1
  3 : 2
  4 : 3
```

Recap.

- $\mathcal{G} \quad \Rightarrow \quad$ Cliques
Separators
- $\mathcal{G}$ decomposable $\quad = \quad$ Triangulated
Only chordless cycles
- $\mathcal{G}$ decomposable $\quad = \quad$ Perfect ordering for vertices
RIP ordering for cliques

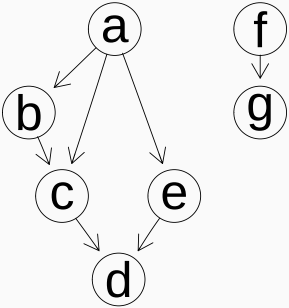
Directed Acyclic Graphs

A Directed Acyclic Graph

- $\vec{\mathcal{G}} = (V, E)$,
  if $\alpha\beta \in E$ then $\beta\alpha \notin E$.
- edges= arrows
- there's no cycles *acyclics*:
  arrows pointing in the same direction all the way around.

A Directed Acyclic Graph

- $\vec{\mathcal{G}} = (V, E)$,
  if $\alpha\beta \in E$ then $\beta\alpha \notin E$.
- edges= arrows
- there's no cycles *acyclics*: *arrows pointing in the same direction all the way around.*
- $V = \{a, b, c, d, e, f, g\}$
- $E = \{ab, ac, ae, bc, cd, ed, fg\}$

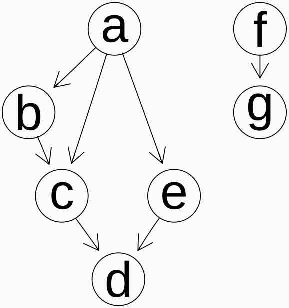


A Directed Acyclic Graph

- If $\alpha\beta \in E$ then α is the parent of β

A Directed Acyclic Graph

- If $\alpha\beta \in E$ then α is the parent of β
- *a* is the parent of *b*
- in R:
    - ~b*a means b is the child of a
    - ~d*c*e means d is the child of c and e



A Directed Acyclic Graph

```
> dag0 <- dag(~a, ~b*a, ~c*a*b, ~d*c*e, ~e*a, ~g*f)
> dag0 <- dag(~a + b*a + c*a*b + d*c*e + e*a + g*f)
> dag0 <- dag(~a + b|a + c|a*b + d|c*e + e|a + g|f)
> dag0 <- dag("a", c("b","a"), c("c","a","b"), c("d","c","e"),
+             c("e","a"),c("g","f"))
> dag0
A graphNEL graph with directed edges
Number of Nodes = 7
Number of Edges = 7
```

Adjacency matrix

```
> dag0a=dag(~a, ~b*a, ~c*a*b, ~d*c*e, ~e*a, ~g*f,
+             result="matrix")
> dag0a
  a b c d e g f
a 0 1 1 0 1 0 0
b 0 0 1 0 0 0 0
c 0 0 0 1 0 0 0
d 0 0 0 0 0 0 0
e 0 0 0 1 0 0 0
g 0 0 0 0 0 0 0
f 0 0 0 0 0 1 0
```

Path, child, parent

- A path (of length n) from α to β is a sequence of vertices $\alpha = \alpha_0, \ldots, \alpha_n = \beta_n$ such that $\alpha_{i-1} \rightarrow \alpha_i$ is an edge in the graph. If there is a path from α to β we write $\alpha \mapsto \beta$.
- If $\alpha \rightarrow \beta$ α is a *parent* of β and β is a *children* of α.

Path, child, parent

- A path (of length n) from α to β is a sequence of vertices $\alpha = \alpha_0, \ldots, \alpha_n = \beta_n$ such that $\alpha_{i-1} \to \alpha_i$ is an edge in the graph. If there is a path from α to β we write $\alpha \mapsto \beta$.
- If $\alpha \to \beta$ α is a *parent* of β and β is a *children* of α.

```
> parents("d",dag0)
[1] "c" "e"
> children("c",dag0)
[1] "d"
```

Ancestrals

- $\mathrm{an}(\beta) = \{\alpha \in V \text{ such } \alpha \mapsto \beta\}$ *ancestrors* of β
- $A \subseteq V$, $\mathrm{an}(A) = \bigcup_{\beta \in A} \mathrm{an}(\beta)$ *ancestral* set of A.

Ancestrals

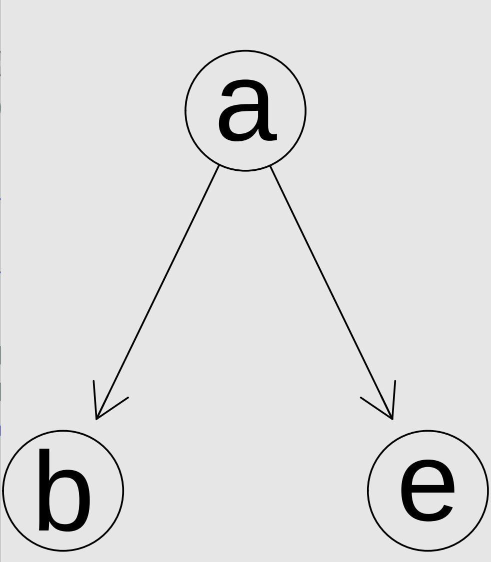
- $\text{an}(\beta) = \{\alpha \in V \text{ such } \alpha \mapsto \beta\}$ *ancestrors* of β
- $A \subseteq V$, $\text{an}(A) = \bigcup_{\beta \in A} \text{an}(\beta)$ *ancestral* set of A.

```
> ancestralSet(c("b","e"),dag0)
[1] "a" "b" "e"
> ancestralGraph(c("b","e"),dag0)
A graphNEL graph with directed edges
Number of Nodes = 3
Number of Edges = 2
> plot(ancestralGraph(c("b","e"),dag0))
```

Ancestrals

- an(β) = {
- $A \subseteq V$, an

```
> ancestra
[1] "a" "b
> ancestra
A graphNEL
Number of 
Number of 
> plot(anc
```

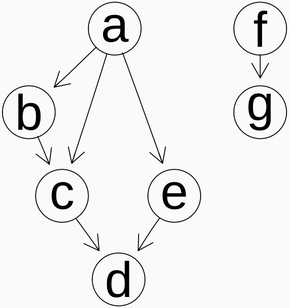


Moralizing a DAG

Moralizing a DAG = Transforming it into an UG (arrows become non-directed) and adding an edge to all parents

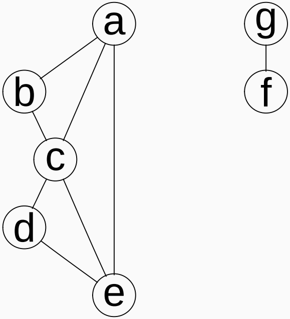
Moralizing a DAG

Moralizing a DAG = Transforming it into an UG (arrows become non-directed) and adding an edge to all parents



Moralizing a DAG

Moralizing a DAG = Transforming it into an UG (arrows become non-directed) and adding an edge to all parents



$$\vec{\mathcal{G}} \longleftrightarrow \mathcal{G}_m$$

$d-$Separation

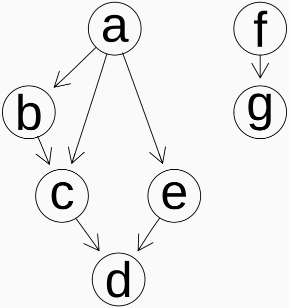
- Let $\alpha \mapsto \beta$ in the DAG $\mathcal{G} = (V, E)$ and $S \subset V$.
- $\alpha \mapsto \beta$ is active according to S if two following conditions hold:
  i. every node with converging edges ($\rightarrow \alpha_i$) is either in S or has a descendant in S,
  ii. every other node is not in S.
- $\alpha \mapsto \beta$ is blocked by S if it is not active according to $\mathcal{G}$
- (A, B, S) three disjoint subsets of V, S $d-$separate A from B if for any path from A to B is blocked by S.

$d-$Separation, Examples

$A = \{a\}, B = \{d\}, S = \{c, e\}$

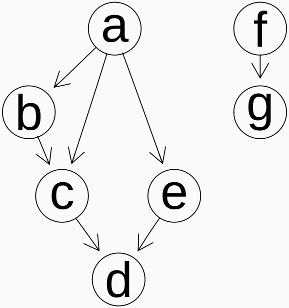
a	$\rightarrow$	c	$\rightarrow$	d	blocked by S
a	$\rightarrow$	e	$\rightarrow$	d	blocked by S

Then S $d-$separates A and B.



d–Separation, Examples

$A = \{a\}, B = \{d\}, S = \{c, e\}$



$a \rightarrow c \rightarrow d$ blocked by S
$a \rightarrow e \rightarrow d$ blocked by S

Then S d–separates A and B.

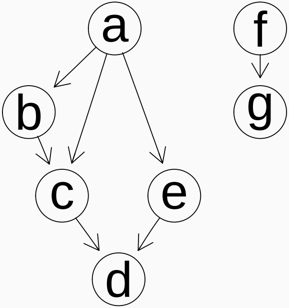
```
> library(ggm)
> dSep(amat = as(dag0, "matrix"),
+       first = "a", second = "d",cond =  c("c","e"))
[1] TRUE
```

$d-$Separation, Examples

$A = \{b\}, B = \{e\}, S = \{c, d\}$

No paths btw A and B
can be blocked by S

Then S doesn't $d-$separate A and B.

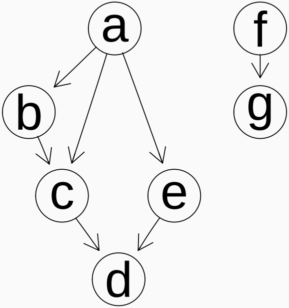


$d-$Separation, Examples

$A = \{b\}, B = \{e\}, S = \{c, d\}$

No paths btw A and B
can be blocked by S

Then S doesn't $d-$separate A and B.




```
> dSep(amat = as(dag0, "matrix"),
+       first = "b", second = "e",cond =  c("c","d"))
[1] FALSE
```

$d-$Separation and Moralization

Theorom

Let $\vec{\mathcal{G}}$ be a DAG and $\mathcal{G}_m$ its moral UG associated to $\vec{\mathcal{G}}$.

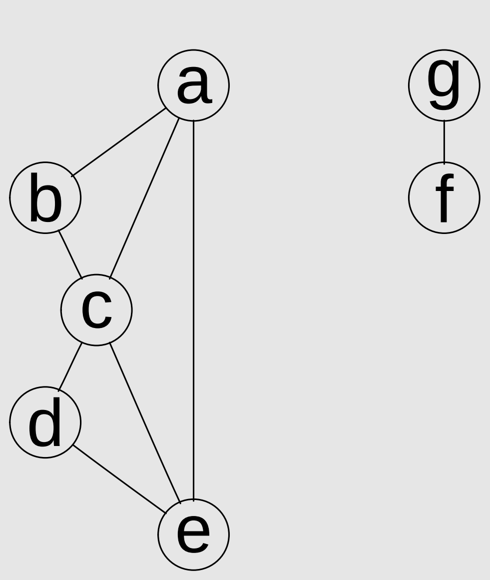
S $d-$separates A and B if and only if S separates A and B in the sub-graph deduced from $\mathcal{G}_m$.

Moralization with R

```
> dag0m <- moralize(dag0)
> dag0m
A graphNEL graph with undirected edges
Number of Nodes = 7
Number of Edges = 8
> plot(dag0m)
```

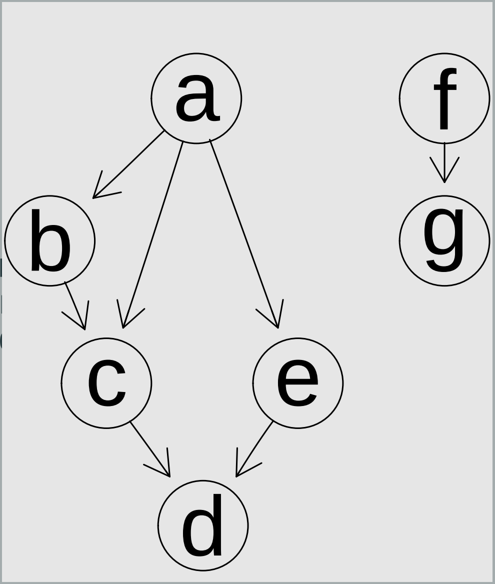
Moralization with R

```
> dag0m <-
> dag0m
A graphNEL
Number of 
Number of 
> plot(dag0
```



Moralization with R

```
> dag0m <-
> dag0m
A graphNEL
Number of
Number of
> plot(dag
```



Markov properties

Conditional Independence

- $\mathrm{X}_V = (X_v, v \in V) \sim P$ a random vector ($\in \mathbb{R}^{|V|}$)
- For $A \subseteq V$, $\mathrm{X}_A = (X_v, v \in A)$
- for all $A, B, S \subseteq V$, $A \perp\!\!\!\perp B \mid S$ means that $\mathrm{X}_A \perp\!\!\!\perp \mathrm{X}_B \mid \mathrm{X}_S$.
- If $f(.)$ is the generic density

$$
\begin{aligned}
A \perp\!\!\!\perp B \mid S \quad &\iff \quad f(x_A, x_B \mid x_c) = f(x_A \mid x_S)\, f(x_B \mid x_S) \\
&\iff \quad f(x_A, x_B, x_S) = h(x_A, x_S)\, g(x_B, x_S)
\end{aligned}
$$

Markov properties for UG

$\mathcal{G} = (V, E)$ is an undirected graph.

(P) We say that P is pairwise Markov w.r.t $\mathcal{G}$, if

$$\alpha \not\sim_{\mathcal{G}} \beta \;\Rightarrow\; \alpha \perp\!\!\!\perp \beta \mid V \setminus \{\alpha, \beta\}$$

(G) We say that P is global Markov w.r.t. $\mathcal{G} = (V, E)$,

$$S \text{ separates } A \text{ and } B \text{ in } \mathcal{G} \;\Rightarrow\; \mathsf{X}_A \perp\!\!\!\perp \mathsf{X}_B \mid \mathsf{X}_S$$

(F) If P has a density f, $\mathcal{C}$ is the set of cliques of $\mathcal{G}$, we say that P factorized Markov w.r.t $\mathcal{G}$, then

$$f(x_V) = \prod_{c \in \mathcal{C}} g_c(x_c)$$

.

Markov properties for UG

$\mathcal{G} = (V, E)$ is an undirected graph.

(P) We say that P is pairwise Markov w.r.t $\mathcal{G}$, if

$$\alpha \not\sim_{\mathcal{G}} \beta \;\Rightarrow\; \alpha \perp\!\!\!\perp \beta \mid V \setminus \{\alpha, \beta\}$$

(G) We say th ... X_S

Theorem

if P has a density f, then

$$(F) \iff (G) \iff (P)$$

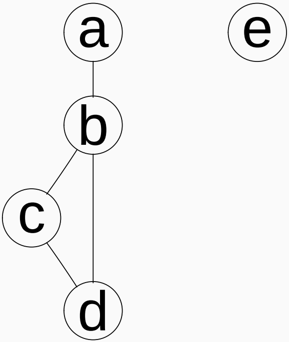
(F) If P has a ... say that P factorized Markov w.r.t $\mathcal{G}$, then

$$f(x_V) = \prod_{c \in \mathcal{C}} g_c(x_c)$$

.

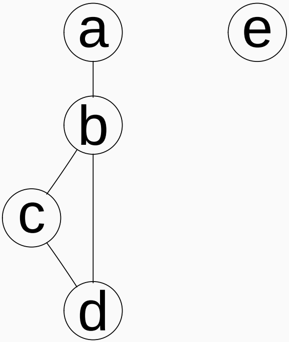
Examples

- $a \perp\!\!\!\perp c \mid b$



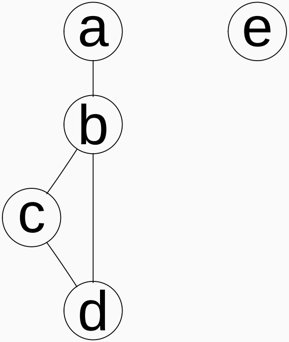
Examples

- $a \perp\!\!\!\perp c \mid b$
- $a \perp\!\!\!\perp e$



Examples

- $a \perp\!\!\!\perp c \mid b$
- $a \perp\!\!\!\perp e$
- $a \perp\!\!\!\perp d \mid b, c.$



Markov properties for DAGs

$\vec{\mathcal{G}} = (V, E)$ is a directed acyclic graph.

(Fd) We say that P admits a recusive factorisation according to $\vec{\mathcal{G}}$ if

$$f(x_V) = \prod_{c \in \mathcal{C}} g_c(x_c \mid x_{\mathrm{pa}(c)})$$

(Gd) P obeys to the directed global Markov property w.r.t $\vec{\mathcal{G}}$

$$S\ d-\text{separates } A \text{ and } B \text{ in } \mathcal{G} \;\Rightarrow\; \mathsf{X}_A \perp\!\!\!\perp \mathsf{X}_B \mid \mathsf{X}_S$$

(Pd) P obeys to the directed pairwise Markov property w.r.t $\vec{\mathcal{G}}$ if

$$\alpha \not\sim_{\mathcal{G}} \beta \;\Rightarrow\; \alpha \perp\!\!\!\perp \beta \mid \mathrm{nd}(\alpha) \setminus \{\beta\}$$

$\mathrm{nd}(\alpha) = V \setminus \mathrm{desc}(\alpha)$ where

$$\mathrm{desc}(\alpha) = \{\beta \in V,\ \alpha \mapsto \beta\}$$

$\vec{\mathcal{G}} = (V, E)$ is a directed acyclic graph.

(Fd) We say that P admits a recusive factorisation according to $\vec{\mathcal{G}}$ if

$$f(x_V) = \prod_{c \in \mathcal{C}} g_c(x_c \mid x_{\text{pa}(c)})$$

(Gd) P obeys t… t $\vec{\mathcal{G}}$

… $_B \mid X_S$

Theorem

if P has a density f, then

$$(Fd) \iff (Gd) \iff (Pd)$$

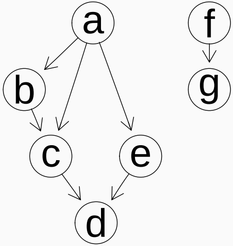
(Pd) P obeys t… w.r.t $\vec{\mathcal{G}}$ if

$$\alpha \not\to_{\mathcal{G}} \beta \;\Rightarrow\; \alpha \perp\!\!\!\perp \beta \mid \text{nd}(\alpha) \setminus \{\beta\}$$

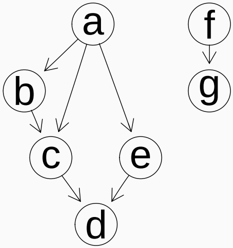
$\text{nd}(\alpha) = V \setminus \text{desc}(\alpha)$ where

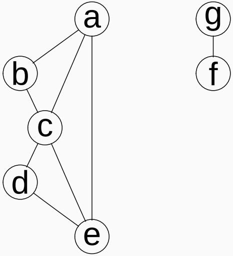
$$\text{desc}(\alpha) = \{\beta \in V,\ \alpha \mapsto \beta\}$$

Examples

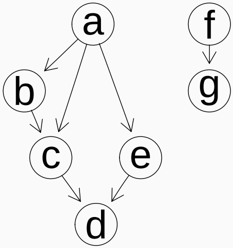


Examples

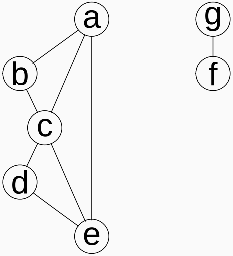




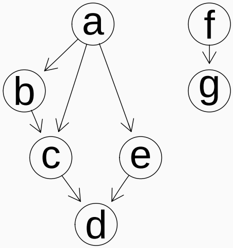
Examples



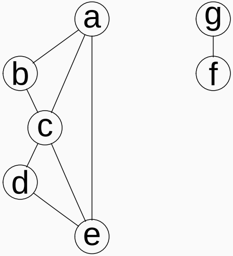
- $a \perp\!\!\!\perp d \mid \{b, c, e\}$



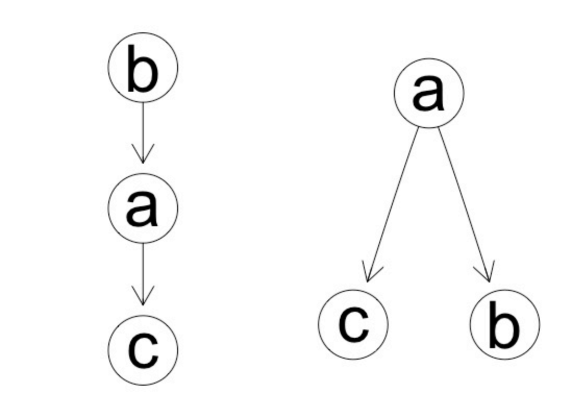
Examples



- $a \perp\!\!\!\perp d \mid \{b, c, e\}$
- $b \perp\!\!\!\perp d \mid \{a, c, e\}$



Markov equivalence



$$b \perp\!\!\!\perp c \mid a$$

Graphical Model

- $\mathsf{X}_V = (X_v,\, v \in V) \sim P$ a random vector ($\in \mathbb{R}^{|V|}$):

$$\mathcal{I}(P) = \{(A, B, S) \subset V \text{ such that } A \perp\!\!\!\perp B \mid S\}$$

Graphical Model

- $\mathbf{X}_V = (X_v,\, v \in V) \sim P$ a random vector ($\in \mathbb{R}^{|V|}$):

$$\mathcal{I}(P) = \{(A, B, S) \subset V \text{ such that } A \perp\!\!\!\perp B \mid S\}$$

- If UG, $\mathcal{G} = (V, E)$:

$$\mathcal{S}(G) = \{(A, B, S) \subset V \text{ such that } S \text{ separates } A \text{ and } B \text{ in } \mathcal{G}\}$$

- $(P, \mathcal{G})$ is a Graphical Model then $\mathcal{S}(\mathcal{G}) \subseteq \mathcal{I}(P)$

Graphical Model

- $\mathsf{X}_V = (X_v,\, v \in V) \sim P$ a random vector ($\in \mathbb{R}^{|V|}$):

$$\mathcal{I}(P) = \{(A, B, S) \subset V \text{ such that } A \perp\!\!\!\perp B \mid S\}$$

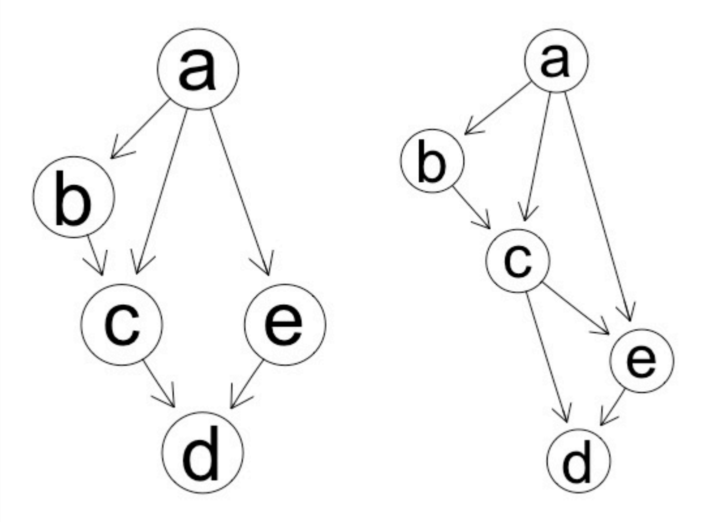
- If DAG, $\vec{\mathcal{G}} = (V, E)$:

$$\mathcal{S}(\vec{\mathcal{G}}) = \{(A, B, S) \subset V \text{ such that } S\, d - \text{separates } A \text{ and } B \text{ in } \mathcal{G}\}$$

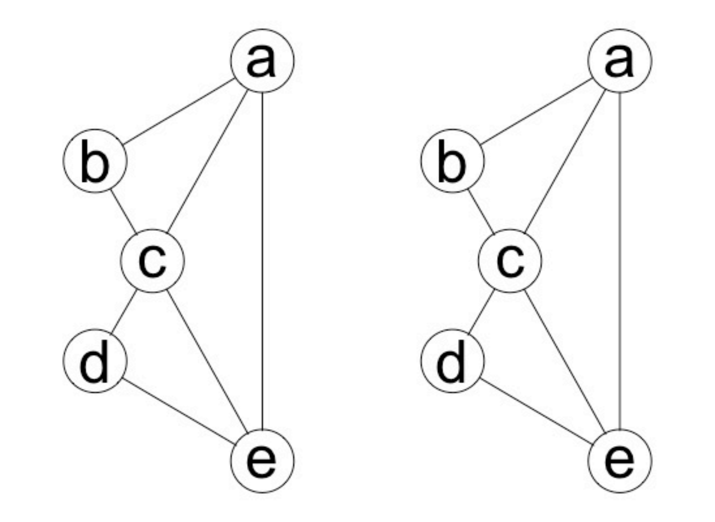
- $(P, \vec{\mathcal{G}})$ is a Graphical Model then $\mathcal{S}(\vec{\mathcal{G}}) \subseteq \mathcal{I}(P)$
- Two DAG $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov Equivalence if

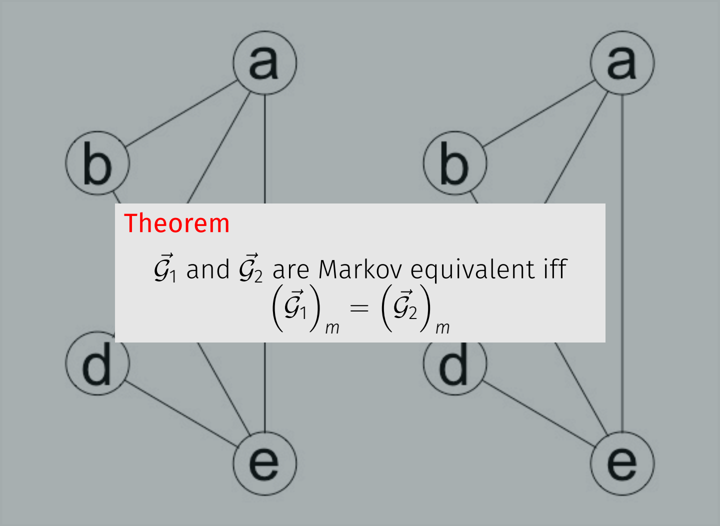
$$\mathcal{S}(\vec{\mathcal{G}}_1) = \mathcal{S}(\vec{\mathcal{G}}_2)$$

Example, Markov equivalence



Example, Markov equivalence





Theorem

$\vec{\mathcal{G}}_1$ and $\vec{\mathcal{G}}_2$ are Markov equivalent iff

$$\left(\vec{\mathcal{G}}_1\right)_m = \left(\vec{\mathcal{G}}_2\right)_m$$

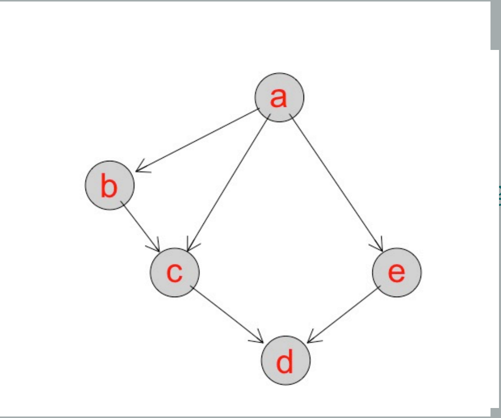
Visualizing graphs with R

Example 1, Customizing the graph

```
> plot(dag0,
+ attrs=list(node = list(fillcolor="lightgrey",
+ fontcolor="red")))
```

Example 1, Customizing the graph

```
> plot(dag(
+ attrs=li                                    grey",
+ fontcolo
```



Example 1, Transforming the graph en LaTeX

```
> library(tikzDevice)
> tikz("g.tex",standAlone = T)
> plot(dag0,
+ attrs=list(node = list(fillcolor="lightgrey",
+ fontcolor="red")))
> dev.off()
```

Example 1, Transforming the graph en LaTeX

```
> library(tikzDevice)
> tikz("g.tex",standAlone = T)
> plot(dag0,
+ attrs=list(node = list(fillcolor="lightgrey",
+ fontcolor="red")))
> dev.off()
```

```
% Created by tikzDevice version 0.10.1 on 2017-03-31 14:50:06
% !TEX encoding = UTF-8 Unicode
\documentclass[10pt]{article}
\usepackage{tikz}

\usepackage[active,tightpage,psfixbb]{preview}

\PreviewEnvironment{pgfpicture}

\setlength\PreviewBorder{0pt}
\begin{document}

\begin{tikzpicture}[x=1pt,y=1pt]
\definecolor{fillColor}{RGB}{255,255,255}
\path[use as bounding box,fill=fillColor,fill opacity=0.00] (0,0) rectangle (505.89,505.89);
\begin{scope}
```

Example 2, with Mixed edges,

Step 1: Construct the adjacency matrix

```
> d1 <- matrix(0,11,11)
> d1[1,2] <- d1[2,1] <- d1[1,3] <- d1[3,1] <- d1[2,4] <- d1[4,2] <-
+ d1[5,6] <- d1[6,5] <- 1
> d1[9,10] <- d1[10,9] <- d1[7,8] <- d1[8,7] <- d1[3,5] <-
+ d1[5,10] <- d1[4,6] <- d1[4,7] <- 1
> d1[6,11] <- d1[7,11] <- 1
> rownames(d1) <- colnames(d1) <- letters[1:11]
> d1
  a b c d e f g h i j k
a 0 1 1 0 0 0 0 0 0 0 0
b 1 0 0 1 0 0 0 0 0 0 0
c 1 0 0 0 1 0 0 0 0 0 0
d 0 1 0 0 0 1 1 0 0 0 0
e 0 0 0 0 0 1 0 0 0 1 0
f 0 0 0 0 1 0 0 0 0 0 1
g 0 0 0 0 0 0 0 1 0 0 1
h 0 0 0 0 0 0 1 0 0 0 0
i 0 0 0 0 0 0 0 0 0 1 0
j 0 0 0 0 0 0 0 0 1 0 0
k 0 0 0 0 0 0 0 0 0 0 0
```

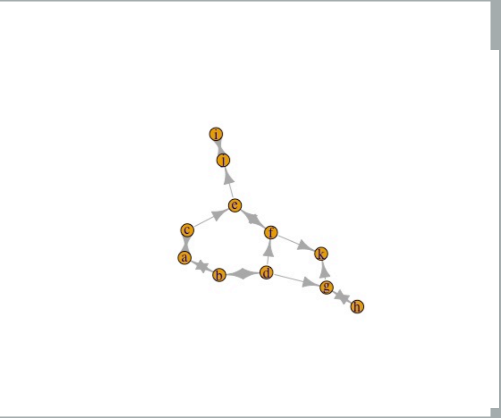
Example 2, with Mixed edges,

Step 2: Transform the adjacency matrix into `igraph` object.

```
> cG1 <- as(d1, "igraph")
> plot(cG1)
```

Step 2: Transf ect.

```
> cG1 <- a
> plot(cG1
```



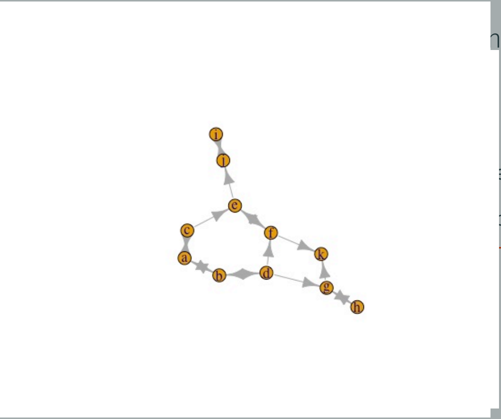
Example 2, with Mixed edges,

Step 3: Changing the type of the edges (only directed and undirected edges

```
> E(cG1)
+ 18/18 edges (vertex names):
 [1] a->b a->c b->a b->d c->a c->e d->b d->f d->g e->f e->j f->e f->k g->h
[15] g->k h->g i->j j->i
> is.mutual(cG1) ##  checks the reciproc pair of the supplied edges
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE
[12]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
> ## Change the bidirected edges to undirected edges
> E(cG1)$arrow.mode <- c(2,0)[1+is.mutual(cG1)]
> plot(cG1, layout=layout.spring)
```

Step 3: Chang nd undirected edges

```
> E(cG1)
+ 18/18 edges
 [1] a->b a->c                                    e->j f->e f->k g->h
[15] g->k h->g
> is.mutual(cG                                     plied edges
 [1]  TRUE  TR                                    LSE  TRUE FALSE
[12]  TRUE FAL
> ## Change th
> E(cG1)$arrow
> plot(cG1, la
```



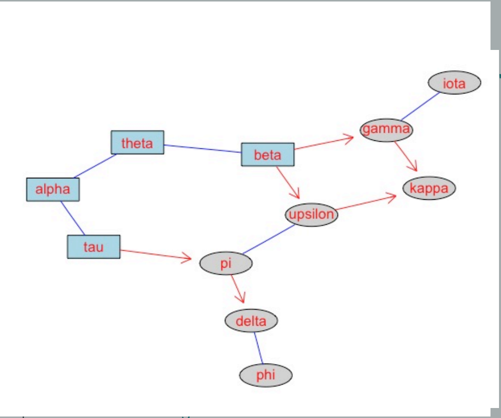
Example 2, with Mixed edges,

Step 4: Renaming and reshaping nodes, Recoloring edges according to the type

```
> cG1a <- as(cG1, "graphNEL")
> nodes(cG1a)
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k"
> nodes(cG1a) <- c("alpha","theta","tau","beta","pi","upsilon","gamma",
+                  "iota","phi","delta","kappa")
> edges <- buildEdgeList(cG1a)
> for (i in 1:length(edges)) {
+   if (edges[[i]]@attrs$dir=="both") {
+     edges[[i]]@attrs$dir <- "none"
+     edges[[i]]@attrs$color <- "blue"
+   }
+    if (edges[[i]]@attrs$dir=="forward") {
+     edges[[i]]@attrs$color <- "red"
+   }
+   }
>  nodes <- buildNodeList(cG1a)
>  for (i in 1:length(nodes)) {
+  nodes[[i]]@attrs$fontcolor <- "red"
+  nodes[[i]]@attrs$shape <- "ellipse"
+  nodes[[i]]@attrs$fillcolor <- "lightgrey"
+  if (i <= 4) {
+  nodes[[i]]@attrs$fillcolor <- "lightblue"
+  nodes[[i]]@attrs$shape <- "box"
+   }
+   }
> cG1al <- agopen(cG1a, edges=edges, nodes=nodes, name="cG1a",
+ layoutType="neato")
> plot(cG1al)
```

Example 2, with Mixed edges,

Step 4: Renaming and reshaping nodes, Recoloring edges according to the type




```
> cG1a <- as(cG1,
> nodes(cG1a)
 [1] "a"  "b"  "c"
> nodes(cG1a) <-                                                   ",
+
> edges <- buildE
> for (i in 1:len
+  if (edges[[i]]
+    edges[[i]]@a
+    edges[[i]]@a
+  }
+   if (edges[[i]
+    edges[[i]]@a
+  }
+  }
>  nodes <- build
>  for (i in 1:le
+  nodes[[i]]@att
+  nodes[[i]]@att
+  nodes[[i]]@att
+  if (i <= 4) {
+  nodes[[i]]@att
+  nodes[[i]]@attrs$shape <- "box"
+  }
+  }
> cG1al <- agopen(cG1a, edges=edges, nodes=nodes, name="cG1a",
+ layoutType="neato")
> plot(cG1al)
```