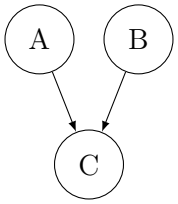
Rapport TP1 INF8225

Elliot Sisteron (1807165)

Question 1

1. Explaining away

Nous allons d'abord expliquer le phénomène, puis nous choisirons un exemple concret. Considérons le réseau bayésien suivant :



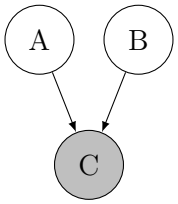
La loi conjointe résultante de ce réseau est la suivante :

$$P(A, B, C) = P(A)P(B)P(C|A, B)$$

On peut clairement voir qu'avant d'avoir plus d'informations, A et B sont indépendantes :

$$\begin{aligned} P(A, B) &= \sum_C P(A, B, C) \\ &= \sum_C P(A)P(B)P(C|A, B) \\ &= P(A)P(B) \sum_C P(C|A, B) \\ &= P(A)P(B) \end{aligned}$$

Supposons maintenant que nous ayons plus d'information sur C.



Vu que C est dépendante de A et de B, cela signifie que l'on peut expliquer cette nouvelle information par A et B. Les lois de A et B vont donc possiblement être modifiée pour expliquer ce phénomène.

D'après notre modèle, il y a clairement une dépendance entre A|C et B|C :

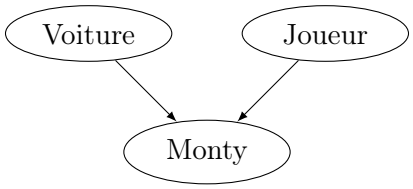
$$P(A, B|C) = \frac{P(C|A, B)P(A, B)}{P(C)} \neq P(A|C)P(B|C)$$

Ainsi, dans cette situation, toute modification de la loi de A ou de B va avoir une répercussion directe sur l'autre variable ; ces deux lois vont être liées pour « expliquer » C.

Passons maintenant à notre exemple. Nous allons considérer le paradoxe de Monty Hall, qui montre bien que le raisonnement bayésien peut être contre-intuitif. Monty Hall était le présentateur du jeu télévisé américain *Let's make a deal*. Dans ce jeu, le candidat est mis face à trois portes. Derrière une des portes se trouve une voiture et derrière les deux autres se trouve une chèvre.

Le candidat doit donc choisir une porte parmi les portes 1, 2 et 3. Une fois ce choix fait, le présentateur ouvre une porte qui n'est pas celle choisie par le joueur et derrière laquelle se trouve une chèvre. Il lui propose ensuite de reconsidérer son choix.

Mettons cela sous la forme d'un réseau bayésien :



La première variable *Voiture* représente la porte derrière laquelle se trouve le prix. *Joueur* représente le choix initial du joueur, avant que le présentateur ouvre la porte. Ces deux variables sont ici bien indépendantes : sans information supplémentaire, le joueur n'a aucun moyen de savoir où se trouve la voiture et donc son choix ne dépend pas de là où se trouve la voiture. Inversement, l'équipe technique du jeu n'a aucun moyen de prédire à l'avance le choix du joueur et de positionner la voiture en fonction de ce choix.

La variable *Monty* décrit le choix du présentateur. Ce choix dépend fortement de deux choses : là où se trouve la voiture (Monty ne veut pas ouvrir cette porte) et le choix du joueur (Monty ne va pas non plus ouvrir la porte du joueur).

Écrivons maintenant les lois de ces v.a. Commençons par *Voiture* et *Joueur*.

v	$P(Voiture = v)$
1	1/3
2	1/3
3	1/3

j	$P(Joueur = j)$
1	1/3
2	1/3
3	1/3

C'est une loi uniforme, sans information complémentaire c'est l'explication la plus rationnelle de la situation : la voiture a une chance sur trois d'être derrière chaque porte. De même, le joueur choisira chaque porte de manière uniforme.

Passons à la loi de *Monty*, c'est là que l'on se rend compte que son choix est très dépendant des variables *Voiture* et *Joueur*.

v	j	$P(Monty = 1\|v, j)$	$P(Monty = 2\|v, j)$	$P(Monty = 3\|v, j)$
1	1	0	1/2	1/2
1	2	0	0	1
1	3	0	1	0
2	1	0	0	1
2	2	1/2	0	1/2
2	3	1	0	0
3	1	0	1	0
3	2	1	0	0
3	3	1/2	1/2	0

Pour expliquer cette table, il suffit de se rendre compte que le choix de Monty Hall est en quelque sorte « forcé » par le choix du joueur et de la position de la voiture. En effet, si le joueur choisit une porte contenant une chèvre, alors comme Monty n'ira pas ouvrir ni la porte du joueur, ni la porte de la voiture, il lui reste un seul choix. Dans le cas où le joueur choisit la porte contenant la voiture, Monty peut ouvrir une des deux portes restantes de manière équiprobable.

On utilise maintenant PMTK sur MatLab pour faire les calculs des probabilités jointes. Voici le code utilisé :

```
% Variables pour plus de lisibilite
% V <-> Voiture, J <-> Joueur, M <-> Monty
V=1;J=2;M=3;
nvars=3;

names=cell(1,3);
names{V}='Voiture';
names{J}='Joueur';
names{M}='Monty';

PORTE1 = 1;
PORTE2 = 2;
PORTE3 = 3;

% Creation du diagramme
dgm = zeros(3,3);
dgm(V,M) = 1;
dgm(J,M) = 1;

% Creation des lois des v.a.
CPDs = cell(3, 1);
CPDs{V} = tabularCpdCreate([0.333; 0.333; 0.333]);
CPDs{J} = tabularCpdCreate([0.333; 0.333; 0.333]);
CPDs{M} = tabularCpdCreate(reshape([0.0 0.0 0.0 0.0 0.5 1.0 0.0 1.0 0.5 0.5 ...
    0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.5 0.5 1.0 0.0 1.0 0.5 0.0 0.0 0.0 0.0], ...
    3, 3, 3));
```

```
26
27 % Calcul des lois jointes
28 diag = dgmCreate(dgm, CPDs, 'nodeNames', names, 'infEngine', 'jtree');
29 joint = dgmInferQuery(diag, [V J M]);
30
31 % Loi de V sachant M = 1
32 clamped = sparsevec(M, PORTE3, nvars);
33 pVgivenM = tabularFactorCondition(joint, V, clamped);
34 fprintf('P(V=1|M=3)=%5.3f\n', pVgivenM.T(PORTE1))
35 fprintf('P(V=2|M=3)=%5.3f\n', pVgivenM.T(PORTE2))
36 fprintf('P(V=3|M=3)=%5.3f\n', pVgivenM.T(PORTE3))
37
38 % Loi de V sachant M = 1 et J = 1
39 clamped = sparsevec([M, J], [PORTE3, PORTE1], nvars);
40 pVgivenMandJ = tabularFactorCondition(joint, V, clamped);
41 fprintf('P(V=1|M=3,J=1)=%5.3f\n', pVgivenMandJ.T(PORTE1))
42 fprintf('P(V=2|M=3,J=1)=%5.3f\n', pVgivenMandJ.T(PORTE2))
43 fprintf('P(V=3|M=3,J=1)=%5.3f\n', pVgivenMandJ.T(PORTE3))
```

Cela nous affiche les valeurs numériques suivantes :

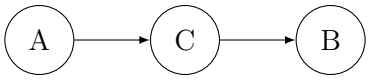
```
P(V=1|M=3)=0.500
P(V=2|M=3)=0.500
P(V=3|M=3)=0.000
P(V=1|M=3,J=1)=0.333
P(V=2|M=3,J=1)=0.667
P(V=3|M=3,J=1)=0.000
```

On remarque bien qu'ici la variable *Voiture* est dépendante de la variable *Joueur* lorsque *Monty* est fixé. C'est ce phénomène que l'on appelle « explaining away » : la loi de *Voiture* va s'adapter à la nouvelle situation (i.e. le *Joueur* qui choisit la porte 1) pour expliquer le fait que *Monty* ait choisi la porte 3.

Dans cette situation, c'est très paradoxal car intuitivement, une fois que Monty a ouvert une porte on s'imaginerait que la voiture se cache derrière les deux portes restantes de manière équiprobable. Or, le fait d'avoir cette information du présentateur nous permet en fait de nous rendre compte qu'il est bien plus intéressant de changer de porte (probabilité de succès de 2/3 vs. 1/3 si on garde la porte initiale).

2. Serial blocking

Le phénomène de « serial blocking » peut s'expliquer en considérant un réseau bayésien du type :



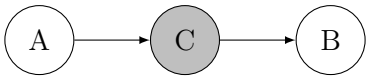
Sa loi conjointe est :

$$P(A, B, C) = P(A)P(B|C)P(C|A)$$

Ici, A et B sont dépendantes :

$$\begin{aligned} P(A, B) &= \sum_C P(A, B, C) \\ &= P(A) \sum_C P(B|C)P(C|A) \\ &\neq P(A)P(B) \end{aligned}$$

Un changement dans A se répercutera sur B, et inversement.
Fixons maintenant C :



Dans ce cas de figure, on peut écrire :

$$\begin{aligned} P(A, B|C) &= \frac{P(A, B, C)}{P(C)} \\ &= \frac{P(A)P(B|C)P(C|A)}{P(C)} \\ &= \frac{P(A)P(B, C)P(C, A)}{P(A)\left(P(C)\right)^2} \\ &= \frac{P(B, C)P(C, A)}{P(C)^2} \\ &= P(A|C)P(B|C) \end{aligned}$$

Ainsi, A et B deviennent conditionnellement indépendantes sachant C. I.e. les changements qui pouvaient avoir des répercussions auparavant n'en ont plus. On dira donc que C fait un « serial blocking ».

Prenons un exemple pour montrer cette situation. Considérons un joueur de football pendant un jour de l'année. Selon le jour, il peut être amené à jouer un match. Durant ce match, ce joueur peut éventuellement se blesser. Et, selon la gravité de la blessure, cela peut l'amener à être hospitalisé.

On peut interpréter cette situation sous forme de réseau bayésien :

Après avoir collecté des données sur notre population cible, on arrive à aux lois suivantes :

m	$P(M = m)$
V	50/365
F	315/365

m	$P(B = V\|m)$	$P(B = F\|m)$
V	1/10	9/10
F	1/500	499/500

b	$P(H = V\|b)$	$P(H = F\|b)$
V	7/10	3/10
F	1/1000	999/1000

Passons au code PMTK :

```
% Variables pour plus de lisibilite
% M <-> Match, B <-> Blessure, H <-> Hopital
M=1;B=2;H=3;
nvars=3;

names=cell(1,3);
names{C}='Match';
names{T}='Blessure';
names{A}='Hopital';

VRAI = 1;
FAUX = 2;

% Creation du diagramme
dgm = zeros(3,3);
dgm(M,B) = 1;
dgm(B,H) = 1;

% Creation des lois des v.a.
CPDs = cell(3, 1);
CPDs{M} = tabularCpdCreate([0.137; 0.863]);
CPDs{B} = tabularCpdCreate(reshape([0.1 0.002 0.9 0.998], 2, 2));
CPDs{H} = tabularCpdCreate(reshape([0.7 0.001 0.3 0.999], 2, 2));

% Calcul des lois jointes
diag = dgmCreate(dgm, CPDs, 'nodeNames', names, 'infEngine', 'jtree');
joint = dgmInferQuery(diag, [M B H]);

% Loi de M sachant H = VRAI
clamped = sparsevec(H, VRAI, nvars);
pMgivenH = tabularFactorCondition(joint, M, clamped);
fprintf('P(M=F|H=V)=%5.3f\n', pMgivenH.T(FAUX))
fprintf('P(M=V|H=V)=%5.3f\n', pMgivenH.T(VRAI))

% Loi de M sachant B = VRAI
clamped = sparsevec(B, VRAI, nvars);
pMgivenB = tabularFactorCondition(joint, M, clamped);
fprintf('P(M=F|B=V)=%5.3f\n', pMgivenB.T(FAUX))
fprintf('P(M=V|B=V)=%5.3f\n', pMgivenB.T(VRAI))

% Loi de M sachant H = VRAI et B = VRAI
```

```
42 clamped = sparsevec([H, B], [VRAI, VRAI], nvars);
43 pMgivenHandB = tabularFactorCondition(joint, M, clamped);
44 fprintf('P(M=F|H=V,B=V)=%5.3f\n', pMgivenHandB.T(FAUX))
45 fprintf('P(M=V|H=V,B=V)=%5.3f\n', pMgivenHandB.T(VRAI))
```

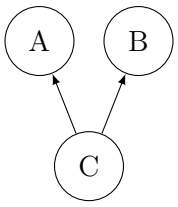
Le résultat est le suivant :

```
P(M=F|H=V)=0.176
P(M=V|H=V)=0.824
P(M=F|B=V)=0.112
P(M=V|B=V)=0.888
P(M=F|H=V,B=V)=0.112
P(M=V|H=V,B=V)=0.888
```

On voit bien que *Match* et *Hopital* sont dépendantes ($P(M|H) \neq P(M)$). Par contre, une fois *Blessure* fixée, *Match* et *Hopital* deviennent indépendantes.

3. Divergent blocking

Expliquons d'abord ce qu'est le « divergent blocking ».



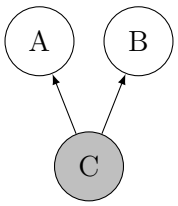
Ce réseau bayésien a la loi conjointe suivante :

$$P(A, B, C) = P(A|C)P(B|C)P(C)$$

On voit clairement que les variables aléatoires A et B sont ici dépendantes :

$$\begin{aligned} P(A, B) &= \sum_C P(A, B, C) \\ &= \sum_C P(A|C)P(B|C)P(C) \\ &= P(A|C)P(B|C) \sum_C P(C) \\ &= P(A|C)P(B|C) \\ &\neq P(A)P(B) \end{aligned}$$

Tout changement de la v.a. A aura donc une répercussion sur B, et vice-versa. Considérons maintenant le cas où C est connue.

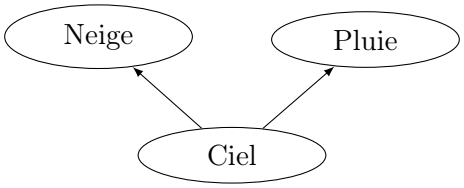


Regardons maintenant ce qu'il se passe :

$$P(A,B|C) = \frac{P(A,B,C)}{P(C)} = \frac{P(A|C)P(B|C)P(C)}{P(C)} = P(A|C)P(B|C)$$

A et B sont donc conditionnellement indépendantes, sachant C. Cela signifie que tout changement de A n'aura plus de répercussion sur C, et inversement. On dira donc que C fait un « divergent blocking ».

Nous allons considérer l'exemple suivant :



Dans cet exemple, la variable aléatoire *Ciel* nous indique si le ciel est gris. Cela peut avoir pour conséquence de produire de la neige ou de la pluie.

Voici les lois associées :

c	$P(C=c)$
V	7/30
F	23/30

c	$P(P=V\|c)$	$P(P=F\|c)$
V	1/2	1/2
F	1/3	2/3

c	$P(N=V\|c)$	$P(N=F\|c)$
V	3/10	7/10
F	1/10	9/10

Le code PMTK est le suivant :

```
% Variables pour plus de lisibilite
% C <-> Ciel, P <-> Pluie, N <-> Neige
C=1;P=2;N=3;
nvars=3;

names=cell(1,3);
names{C}='Ciel';
names{P}='Pluie';
names{N}='Neige';

VRAI = 1;
FAUX = 2;

% Creation du diagramme
dgm = zeros(3,3);
```

```
dgm(C,P) = 1;
dgm(C,N) = 1;

% Creation des lois des v.a.
CPDs = cell(3, 1);
CPDs{C} = tabularCpdCreate([0.23; 0.76]);
CPDs{P} = tabularCpdCreate(reshape([0.5 0.33 0.5 0.66], 2, 2));
CPDs{N} = tabularCpdCreate(reshape([0.3 0.1 0.7 0.9], 2, 2));

% Calcul des lois jointes
diag = dgmCreate(dgm, CPDs, 'nodeNames', names, 'infEngine', 'jtree');
joint = dgmInferQuery(diag, [C P N]);

% Loi de P
p = tabularFactorMarginalize(joint, P);
fprintf('P(P=F)=%5.3f\n', p.T(FAUX))
fprintf('P(P=V)=%5.3f\n', p.T(VRAI))

% Loi de P sachant N = VRAI
clamped = sparsevec(N, VRAI, nvars);
pPgivenN = tabularFactorCondition(joint, P, clamped);
fprintf('P(P=F|N=V)=%5.3f\n', pPgivenN.T(FAUX))
fprintf('P(P=V|N=V)=%5.3f\n', pPgivenN.T(VRAI))

% Loi de P sachant C = VRAI
clamped = sparsevec(C, VRAI, nvars);
pPgivenC = tabularFactorCondition(joint, P, clamped);
fprintf('P(P=F|C=V)=%5.3f\n', pPgivenC.T(FAUX))
fprintf('P(P=V|C=V)=%5.3f\n', pPgivenC.T(VRAI))

% Loi de P sachant C = VRAI et N = VRAI
clamped = sparsevec([C, N], [VRAI, VRAI], nvars);
pPgivenCandN = tabularFactorCondition(joint, P, clamped);
fprintf('P(P=F|C=V,N=V)=%5.3f\n', pPgivenCandN.T(FAUX))
fprintf('P(P=V|C=V,N=V)=%5.3f\n', pPgivenCandN.T(VRAI))
```

Ce qui donne comme résultat :

```
P(P=F)=0.628
P(P=V)=0.372
P(P=F|N=V)=0.587
P(P=V|N=V)=0.413
P(P=F|C=V)=0.500
P(P=V|C=V)=0.500
P(P=F|C=V,N=V)=0.500
P(P=V|C=V,N=V)=0.500
```

P et N sont bien dépendantes, on voit bien que la loi de P n'est pas la même que la loi de $P|N$. On remarque bien aussi phénomène de blocage ; la loi de $P|C$ est la même que la loi de $P|C, N$.

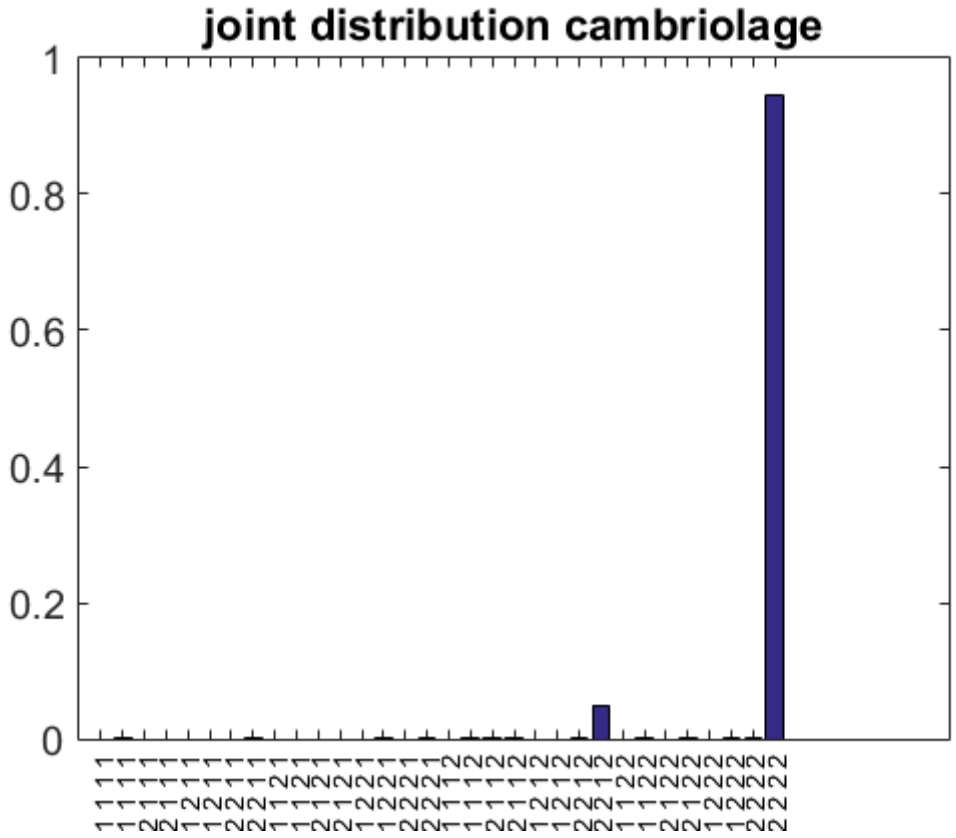
Question 2

1. Code mkDetecterCambriolageDgm.m

```
% Variables pour plus de lisibilite
C=1;T=2;A=3;M=4;J=5;
nvars=5;
names=cell(1,5);
names{C}='Cambriolage';
names{T}='Tremblement';
names{A}='Alarme';
names{M}='MarieAppelle';
names{J}='JeanAppelle';

VRAI = 1;
FAUX = 2;

% Creation du diagramme
dgm = zeros(5,5);
dgm(C,A) = 1;
dgm(T,[A, J]) = 1;
dgm(A,[M, J]) = 1;

CPDs = cell(5, 1);
CPDs{C} = tabularCpdCreate([0.001; 0.999]);
CPDs{T} = tabularCpdCreate([0.002; 0.998]);
CPDs{A} = tabularCpdCreate(reshape([0.95 0.94 0.29 0.001 0.05 0.06 0.71 ...
    0.999], ...
    2, 2, 2));
CPDs{M} = tabularCpdCreate(reshape([0.9 0.05 0.1 0.95], 2, 2));
CPDs{J} = tabularCpdCreate(reshape([0.05 0.001 0.5 0.001 0.95 0.999 0.5 ...
    0.999], ...
    2, 2, 2));

% Loi jointe
diag = dgmCreate(dgm, CPDs, 'nodeNames', names, 'infEngine', 'jtree');
joint = dgmInferQuery(diag, [C T A M J]);

% Histogramme
lab = cellfun(@(x) {sprintf('%d ',x)}, num2cell(ind2subv([2 2 2 2 2],1:32),2));
figure;
bar(joint.T(:))
set(gca,'xtick',1:32);
xticklabelRot(lab, 90, 10, 0.01)
title('joint distribution cambriolage')

% Marginales conditionelles
clamped = sparsevec([M,J], [VRAI, FAUX], nvars);
pCgivenMandJ = tabularFactorCondition(joint, C, clamped);
```

```
44 fprintf('P(C=V|M=V,J=F)=%5.7f\n', pCgivenMandJ.T(VRAI))
45
46 clamped = sparsevec([M,J], [FAUX, VRAI], nvars);
47 pCgivenMandJ = tabularFactorCondition(joint, C, clamped);
48 fprintf('P(C=V|M=F,J=V)=%5.7f\n', pCgivenMandJ.T(VRAI))
49
50 clamped = sparsevec([M,J], [VRAI, VRAI], nvars);
51 pCgivenMandJ = tabularFactorCondition(joint, C, clamped);
52 fprintf('P(C=V|M=V,J=V)=%5.7f\n', pCgivenMandJ.T(VRAI))
53
54 clamped = sparsevec([M,J], [FAUX, FAUX], nvars);
55 pCgivenMandJ = tabularFactorCondition(joint, C, clamped);
56 fprintf('P(C=V|M=F,J=F)=%5.7f\n', pCgivenMandJ.T(VRAI))
57
58 clamped = sparsevec(M, VRAI, nvars);
59 pCgivenM = tabularFactorCondition(joint, C, clamped);
60 fprintf('P(C=V|M=V)=%5.7f\n', pCgivenM.T(VRAI))
61
62 clamped = sparsevec(J, VRAI, nvars);
63 pCgivenJ = tabularFactorCondition(joint, C, clamped);
64 fprintf('P(C=V|J=V)=%5.7f\n', pCgivenJ.T(VRAI))
65
66 % Marginales inconditionelles
67 pC = tabularFactorMarginalize(joint, C);
68 fprintf('P(C=V)=%5.7f\n', pC.T(VRAI))
69
70 pT = tabularFactorMarginalize(joint, T);
71 fprintf('P(T=V)=%5.7f\n', pT.T(VRAI))
72
73 pA = tabularFactorMarginalize(joint, A);
74 fprintf('P(A=V)=%5.7f\n', pA.T(VRAI))
75
76 pM = tabularFactorMarginalize(joint, M);
77 fprintf('P(M=V)=%5.7f\n', pM.T(VRAI))
78
79 pJ = tabularFactorMarginalize(joint, J);
80 fprintf('P(J=V)=%5.7f\n', pJ.T(VRAI))
```

2. Histogramme probabilités jointes



3. Marginales conditionnelles avec PMTK

P(C=V|M=V,J=F)=0.0056560
P(C=V|M=F,J=V)=0.0007491
P(C=V|M=V,J=V)=0.0027700
P(C=V|M=F,J=F)=0.0007414
P(C=V|M=V)=0.0056484
P(C=V|J=V)=0.0009922

4. Marginales inconditionnelles de chaque variable du modèle

P(C=V)=0.0010000
P(T=V)=0.0020000
P(A=V)=0.0031664
P(M=V)=0.0526915
P(J=V)=0.0011520

5. P(J) et P(C|J=V)

On a, en utilisant la loi jointe du réseau :

$$\begin{aligned} P(J) &= \sum_{A,C,T,M} P(A,C,T,M,J) \\ &= \sum_{A,C,T,M} P(C)P(T)P(A|T,C)P(M|A)P(J|T,A) \\ &= \sum_{T} P(T) \sum_{C} P(C) \sum_{A} P(A|T,C)P(J|T,A) \sum_{M} P(M|A) \end{aligned}$$

Pour $P(C|J=V)$, on connait maintenant $P(J=V)$. On peut essayer de le faire par la règle de bayes :

$$P(C|J=V) = \frac{P(J=V|C)P(C)}{P(J=V)}$$

Ou bien directement :

$$\begin{aligned} P(C|J=V) &= \frac{P(C,J=V)}{P(J=V)} \\ &= \frac{P(C)\sum_T P(T)\sum_A P(A|T,C)P(J=V|T,A)\sum_M P(M|A)}{P(J=V)} \end{aligned}$$