# Devoir théorique 1 : Partie R

UNIVERSITÉ DE
SHERBROOKE

Math pour l'intelligence artificielle

STT760

*Étudiants (CIP) :*
Tahar Amairi (amat0601)
Céline Zhang (zhac3201)
Omar Chida (chim2708)
Corentin Pommelec (pomc0601)

*Enseignants :*
Felix Camirand Lemyre
Samuel Valiquette

28 octobre 2022

# Question 1

Avant tout, il faut installer et charger les librairies nécessaires si ce n'est pas déjà fait :

```
# Chargement des librairies
library("gRain", "gRbase", "Rgraphviz")
```

## 1.

Nous cherchons à produire une représentation graphique du réseau bayésien à partir des informations fournies par Claude. Ainsi, nous utiliserons des abréviations, pour chaque noeud, correspondantes aux différentes variables aléatoires renseignées :
- Revenus (Élevé, Moyen, Faibles) -> R
- Actifs (Élevé, Moyen, Faibles) -> Ac
- Ratio dettes vs revenus (Élevé, Moyen, Faible) -> DvsR
- Historique de paiement (Bon,mauvais) -> H
- Âge (moins de 25 ans, entre 25 et 50, entre 50 et 65, 65 et plus) -> Ag
- Fiabilité (Fiable, non fiable) -> Fi
- Revenus futurs (Élevés, Moyens, Faibles) -> Rf
- Solvabilité (Non solvable, Solvable) -> S

Pour la création du graphe, nous utilisons un objet graphNEL que nous appelons dag_solvency. Ce graphe est un réseau bayésien qui va contenir les différents noeuds énoncés précédemment. A l'aide des informations fournies par Claude nous pouvons déduire, notamment grâce aux point 1 à 8, les dépendances qui existent entre les noeuds. Ces dépendances se traduisent dans le graphe sous forme d'arêtes.

Par exemple pour le point 1. : "Un(e) client(e) avec un bon historique de paiement a tendance à être plus fiable". Cela se traduit par le fait que la fiabilité dépend de l'historique de paiemen. On crée donc une arête du noeud historique vers le noeud de fiabilité.

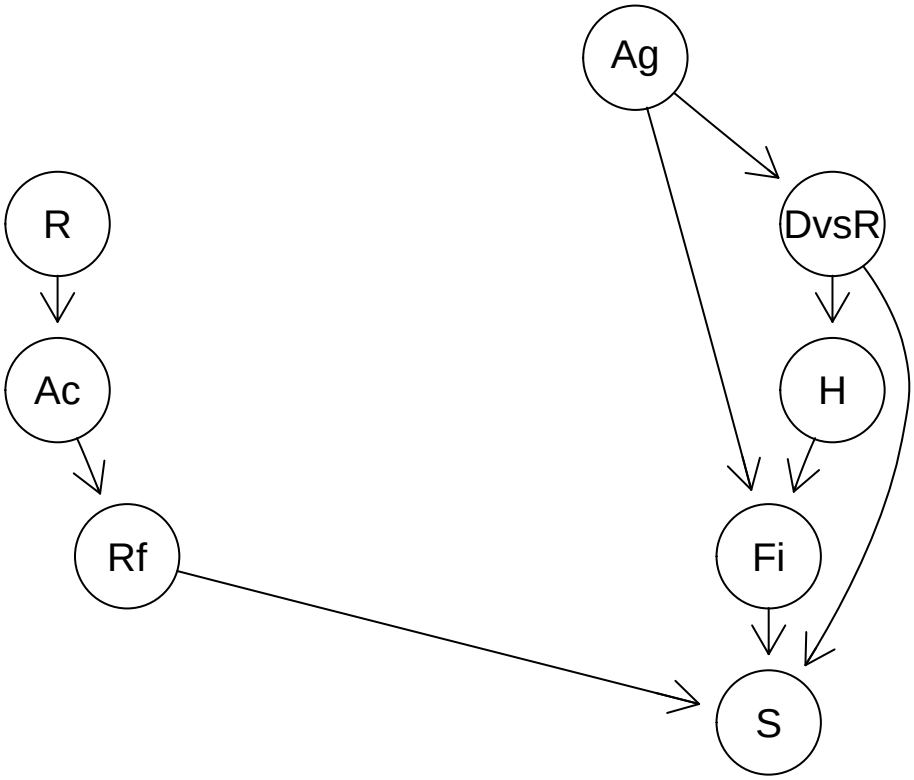
Voici le code R correspondant à la création du réseau bayésien :

```
# Création du grapheNEL
dag_solvency = dag(
  c("R"),
  c("Ac", "R"),
  c("DvsR", "Ag"),
  c("H", "DvsR"),
  c("Ag"),
  c("Fi", "H", "Ag"),
  c("Rf", "Ac"),
  c("S", "Fi", "Rf", "DvsR"),
  result = "graphNEL"
)

# Affichage des caractéristiques du graphe
dag_solvency
```

```
## A graphNEL graph with directed edges
## Number of Nodes = 8
## Number of Edges = 9
```

```r
# Affichage du graphe
plot(dag_solvency)
```



## 2.

Nous souhaitons créer une élicitation des tables de probabilités conditionnelles associées au réseau bayésien créé précédemment. Ces tables de probabilités seront déduites des points 1 à 8, que nous avons en notre possession grâce à l'expérience de Claude.

Par exemple, en tenant compte des abréviations définies précédemment nous pouvons donc déduire des différents points les indications suivantes :
1. Un(e) client(e) avec un bon historique de paiement a tendance à être plus fiable :
P(F = Fiable | H = Bon) > P(F = Fiable | H = Mauvais)
2. Plus un(e) client(e) est âgé(e), plus il/elle a de chance d'être fiable :
P(F = Fiable | Ag = +65) > P(F = Fiable | Ag = -25)
3. Les clients plus âgés ont tendance à avoir un fiable ratio dettes vs revenu :
P(DvsR = Faible | Ag = +65) > P(DvsR = Faible | Ag = -25)
4. La probabilité d'avoir un bon historique de paiement augmente au fur et à mesure que le ratio de dette vs revenus diminue :
P(H = Bon | DvsR = Faible) > P(H = Bon | DvsR = Flevé)
5. Plus les revenus d'une personne sont élevés, plus cette personne a de chance d'avoir des actifs élevés :
P(Ac = Elevé | R = Elevé) > P(Ac = Flevé | R = Faible)
6. Plus une personne a d'actifs, plus cette personne a de chance d'avoir un revenu élevé dans le futur :
P(Rf = Elevé | Ac = Elevé) > P(Rf = Elevé | Ac = Faible)
7. Une personne fiable a tendance à être plus solvable qu'une personne non fiable :
P(S = Solvable | Fi = Fiable) > P(S = Solvable | Fi = Non fiable)
8. Les personnes qui ont des revenus prometteurs ont + de chance d'être solvables que celles dont la perspective des revenus à venir est mauvaise :
P(S = Solvable | Rf = Elevé) > P(S = Solvable | Rf = Faible)

Nous pourrons vérifier très aisément ces exemples via les tables de probabilités que nous créerons.
Pour créer ces tables nous commençons par recréer le réseau bayésien mais cette fois ci en spécifiant les probabilités associées à chaque noeud, à l'aide de la fonction cptable, qui permet de créer les tables de probabilités conditionnelles associées à chaque noeud.
Nous utilisons la fonction cptable avec comme argument le noeud sachant ses parents, les valeurs associées aux tableaux des probabilités conditionnelles à ses parents (indications colonnes par colonnes et tableau par tableau), ainsi que les valeurs possibles du noeuds.

La création d'un objet de type grain (GRAphical Independence Network) nommé grain_solvency va nous permettre à l'aide de la fonction querygrain d'afficher les tables de probabilités conditionnelles associées aux noeuds que l'on spécifie en paramètre.
C'est grâce à ces tables que l'on va pouvoir vérifier que les probabilités que l'on a défini par élicitation des points 1 à 8 sont correctes ou non.

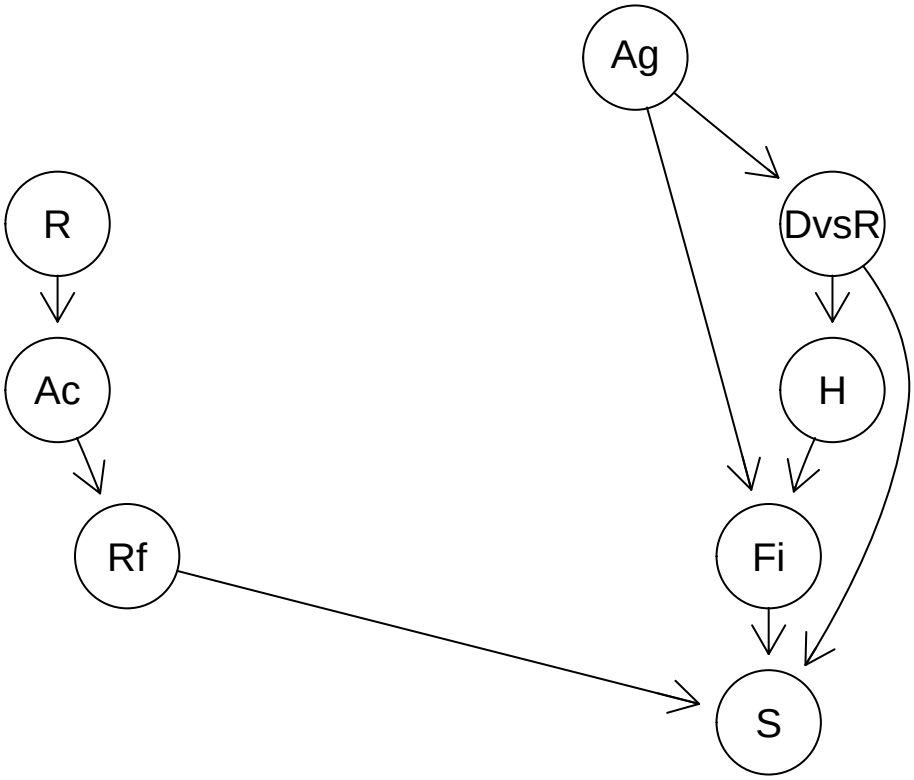
Voici le code R correspondant à l'élicitation des tables de probabilités, respectant les points 1 à 8 susmentionnés :

```
# Création du réseau bayésien en spécifiant les probabilités associés à chaque noeud
val_F_M_E = c("Faible","Moyen", "Elevé")
val_Age = c("-25","25-50", "50-65", "+65")
val_B_M = c("Mauvais", "Bon")
val_F_NF = c("Non fiable", "Fiable")
val_S_NS = c("Non Solvable", "solvable")

cp_R <- cptable(~R,
                values=c(1/3, 1/3, 1/3),
                levels=val_F_M_E)
cp_Ag <- cptable(~Ag,
                 values=c(1/4, 1/4,1/4, 1/4),
                 levels=val_Age)
cp_Ac <- cptable(~Ac|R,
                 values=c(0.8,0.15,0.05, 0.15,0.8,0.05, 0.05,0.15,0.8),
                 levels=val_F_M_E)
cp_Rf <- cptable(~Rf|Ac,
                 values=c(0.7,0.2,0.1, 0.1,0.6,0.3, 0.1,0.2,0.7),
                 levels=val_F_M_E)
cp_DvsR <- cptable(~DvsR|Ag,
                   values=c(0.4,0.3,0.3, 0.5, 0.3, 0.2, 0.6,0.3,0.1, 0.7,0.2,0.1),
                   levels=val_F_M_E)
cp_H <- cptable(~H|DvsR,
                values=c(0.1,0.9, 0.35,0.65, 0.5,0.5),
                levels=val_B_M)
cp_Fi <- cptable(~Fi|Ag+H,
                 values=c(0.6,0.4, 0.3,0.7, 0.2,0.8, 0.1,0.9,
                          0.5,0.5, 0.2,0.8, 0.1,0.9, 0,1),
                 levels=val_F_NF)
cp_S <- cptable(~S|Fi+Rf+DvsR,
                values=c(0.7,0.3, 0.5,0.5,
                         0.6,0.4, 0.4, 0.6,
                         0.5,0.5, 0.3, 0.7,
                         0.6,0.4, 0.4, 0.6,
                         0.6,0.4, 0.3, 0.7,
                         0.5,0.5, 0.2, 0.8,
                         0.7,0.3, 0.4, 0.6,
                         0.5,0.5, 0.2, 0.8,
                         0.4,0.6, 0.1, 0.9),
                levels=val_S_NS)

net_list = compileCPT(list(cp_R,cp_Ag, cp_Ac, cp_Rf, cp_DvsR, cp_H, cp_Fi, cp_S))

# Création d'un objet de type grain (GRAphical Independence Network)
grain_solvency = grain(net_list)
plot(grain_solvency$dag)
```




```
# 1.Un(e) client(e) avec un bon historique de paiement a tendance à être plus fiable :
querygrain(grain_solvency, nodes=c("Fi","H"), type="conditional")
```

```
##             H
## Fi             Mauvais       Bon
##   Non fiable 0.3303665 0.1904762
##   Fiable     0.6696335 0.8095238
```

```
# 2. Plus un(e) client(e) est âgé(e), plus il/elle a de chance d'être fiable :
querygrain(grain_solvency, nodes=c("Fi","Ag"), type="conditional")
```

```
##             Ag
## Fi              -25  25-50  50-65   +65
##   Non fiable 0.5295 0.2255 0.1215 0.019
##   Fiable     0.4705 0.7745 0.8785 0.981
```

```
# 3. Les clients plus âgés ont tendance à avoir un faible ratio dettes vs revenus :
querygrain(grain_solvency, nodes=c("DvsR","Ag"), type="conditional")
```

```
##         Ag
## DvsR     -25 25-50 50-65 +65
##   Faible 0.4   0.5   0.6 0.7
##   Moyen  0.3   0.3   0.3 0.2
##   Elevé  0.3   0.2   0.1 0.1
```

```
# 4. La probabilité d'avoir un bon historique de paiement augmente au fur et à mesure que le ratio de d
querygrain(grain_solvency, nodes=c("H","DvsR"), type="conditional")
```

```
##          DvsR
## H         Faible Moyen Elevé
##   Mauvais    0.1  0.35   0.5
##   Bon        0.9  0.65   0.5
```

```
# 5. Plus les revenus d'une personne sont élevés, plus cette personne a de chance d'avoir des actifs él
querygrain(grain_solvency, nodes=c("Ac","R"), type="conditional")
```

```
##         R
## Ac       Faible Moyen Elevé
##   Faible   0.80  0.15  0.05
##   Moyen    0.15  0.80  0.15
##   Elevé    0.05  0.05  0.80
```

```
# 6. Plus une personne a d'actifs, plus cette personne a de chance d'avoir un revenu élevé dans le futu
querygrain(grain_solvency, nodes=c("Rf","Ac"), type="conditional")
```

```
##         Ac
## Rf       Faible Moyen Elevé
##   Faible    0.7   0.1   0.1
##   Moyen     0.2   0.6   0.2
##   Elevé     0.1   0.3   0.7
```

```
# 7. Une personne fiable a tendance à être plus solvable qu'une personne non fiable :
querygrain(grain_solvency, nodes=c("S","Fi"), type="conditional")
```

```
##               Fi
## S              Non fiable   Fiable
##   Non Solvable  0.5669671 0.342742
##   solvable      0.4330329 0.657258
```

```
# 8. Les personnes qui ont des revenus prometteurs ont + de chance d'être solvables que celles dont la
querygrain(grain_solvency, nodes=c("S","Rf"), type="conditional")
```

```
##               Rf
## S               Faible     Moyen     Elevé
##   Non Solvable 0.50565 0.3951125 0.2951125
##   solvable     0.49435 0.6048875 0.7048875
```

On remarque, grâce aux affichages des tables de probabilités conditionnelles ci-dessus, que toutes les indications fournies par les points 1 à 8 sont respectées.

# Question 2

## 1.

La première étape est d'importer le jeu de données *marks* et les librairies qu'on va utiliser :

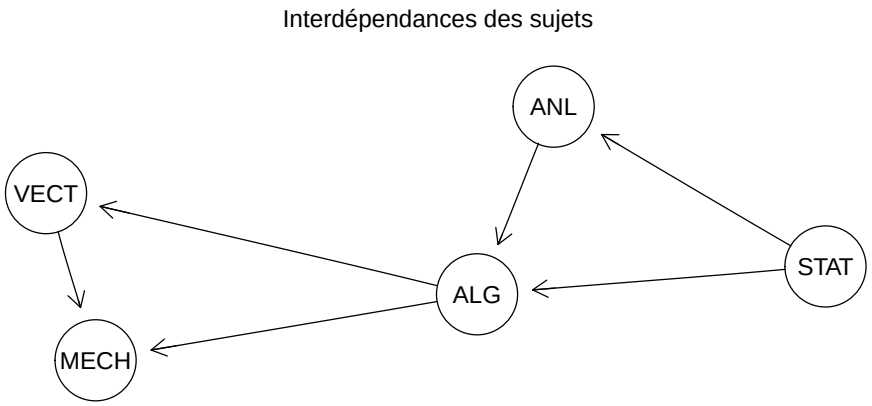
```
# import libs and data
library("bnlearn")
library("Rgraphviz")

data(marks)
dagNotes = empty.graph(names(marks))
arcs(dagNotes) = matrix(c("VECT", "MECH","ALG", "MECH",
                          "ALG", "VECT","ANL", "ALG",
                          "STAT", "ALG","STAT", "ANL"),
                          ncol = 2, byrow = TRUE, dimnames = list(c(), c("from", "to")))

vStructs = list(arcs = vstructs(dagNotes, arcs = TRUE))
```

On peut aussi visualiser la SRB associée :

```
graphviz.plot(dagNotes, highlight = vStructs, layout = "fdp", main = "Interdépendances des sujets")
```




On peut maintenant construire la matrice *notesReussite* : il est important de noter la transformation en facteur de chacune de ses colonnes. En effet, cela est nécessaire pour l'exécution de la fonction *bn.fit* ultérieurement.

```
notesReussite = as.data.frame((marks >= 45) * 1)
notesReussite[notesReussite == 1] = "R"
notesReussite[notesReussite == 0] = "E"

#to set a column as factor
toFactor <- function(x)
{
  as.factor(x)
}

#set factors
notesReussite[] = lapply(notesReussite, toFactor)
```

## 2.

- $f_1(x, p_1) = p_1^x \times (1 - p_1)^{(1-x)}$ avec :
  $\begin{cases} p_1 = \mathbb{P}(X_{i5} = 1) \end{cases}$
- $f_2(x, y, p_2, p_3) = (p_2^x \times (1 - p_2)^{(1-x)})^y \times (p_3^x \times (1 - p_3)^{(1-x)})^{(1-y)}$ avec :
  $\begin{cases} p_2 = \mathbb{P}(X_{i4} = 1 | X_{i5} = 1) \\ p_3 = \mathbb{P}(X_{i4} = 1 | X_{i5} = 0) \end{cases}$
- $f_3(x, y, z, p_4, p_5, p_6, p_7) = (p_4^x \times (1 - p_4)^{(1-x)})^{yz} \times (p_5^x \times (1 - p_5)^{(1-x)})^{(1-y)z} \times (p_6^x \times (1 - p_6)^{(1-x)})^{y(1-z)}$
  $\times (p_7^x \times (1 - p_7)^{(1-x)})^{(1-y)(1-z)}$ avec :
  $\begin{cases} p_4 = \mathbb{P}(X_{i3} = 1 | X_{i4} = 1, X_{i5} = 1) \\ p_5 = \mathbb{P}(X_{i3} = 1 | X_{i4} = 0, X_{i5} = 1) \\ p_6 = \mathbb{P}(X_{i3} = 1 | X_{i4} = 1, X_{i5} = 0) \\ p_7 = \mathbb{P}(X_{i3} = 1 | X_{i4} = 0, X_{i5} = 0) \end{cases}$
- $f_4(x, y, p_8, p_9) = (p_8^x \times (1 - p_8)^{(1-x)})^y \times (p_9^x \times (1 - p_9)^{(1-x)})^{(1-y)}$ avec :
  $\begin{cases} p_8 = \mathbb{P}(X_{i2} = 1 | X_{i3} = 1) \\ p_9 = \mathbb{P}(X_{i2} = 1 | X_{i3} = 0) \end{cases}$
- $f_5(x, y, z, p_{10}, p_{11}, p_{12}, p_{13}) = (p_{10}^x \times (1 - p_{10})^{(1-x)})^{yz} \times (p_{11}^x \times (1 - p_{11})^{(1-x)})^{(1-y)z}$
  $\times (p_{12}^x \times (1 - p_{12})^{(1-x)})^{y(1-z)} \times (p_{13}^x \times (1 - p_{13})^{(1-x)})^{(1-y)(1-z)}$ avec :
  $\begin{cases} p_{10} = \mathbb{P}(X_{i1} = 1 | X_{i2} = 1, X_{i3} = 1) \\ p_{11} = \mathbb{P}(X_{i1} = 1 | X_{i2} = 0, X_{i3} = 1) \\ p_{12} = \mathbb{P}(X_{i1} = 1 | X_{i2} = 1, X_{i3} = 0) \\ p_{13} = \mathbb{P}(X_{i1} = 1 | X_{i2} = 0, X_{i3} = 0) \end{cases}$

Voici maintenant l'implémentation de chacune de ces fonctions :

```
f1 = function(x, p1)
{
  a = 0

  if((x == 0) || (x == 1))
  {
    a = p1^x * (1 - p1)^(1 - x)
  }

  return(a)
}

f2 = function(x, y, p2, p3)
{
  a = 0

  if(((x == 0) || (x == 1)) && ((y == 0) || (y == 1)))
  {
    a = f1(x, p2)^y * f1(x, p3)^(1 - y)
  }

  return(a)
}
```

```
f3 = function(x, y, z, p4, p5, p6, p7)
{
  a = 0

  if(((x == 0) || (x == 1)) && ((y == 0) || (y == 1)) && ((z == 0) || (z == 1)))
  {
    tmp1 = f1(x, p4)^(y*z) * f1(x, p5)^((1 - y)*z)
    tmp2 = f1(x, p6)^((1 - z)*y) * f1(x, p7)^((1 - y)*(1 - z))
    a = tmp1 * tmp2
  }

  return(a)
}

f4 = function(x, y, p8, p9)
{
  return(f2(x, y, p8, p9))
}

f5 = function(x, y, z, p10, p11, p12, p13)
{
  return(f3(x, y, z, p10, p11, p12, p13))
}

# Xi = (x1, x2, ..., x5)
# P = (p1, p2, ..., p13)
L = function(Xi, P)
{
  F1 = f1(Xi[5], P[1])
  F2 = f2(Xi[4], Xi[5], P[2], P[3])
  F3 = f3(Xi[3], Xi[4], Xi[5], P[4], P[5], P[6], P[7])
  F4 = f4(Xi[2], Xi[3], P[8], P[9])
  F5 = f5(Xi[1], Xi[2], Xi[3], P[10], P[11], P[12], P[13])

  return(F1 * F2 * F3 * F4 * F5)
}
```

### 3.

Tout d'abord, on note que $L(X_i, p) = f_1 \times f_2 \times f_3 \times f_4 \times f_5$. Par conséquent, le maximum de vraisemblance s'écrit :

$$
\begin{aligned}
p* &= argmin \prod_{i=1}^{n} L(X_i, p) \\
&= argmin \prod_{i=1}^{n} f_1 \times f_2 \times f_3 \times f_4 \times f_5 \\
&= argmin \ln(\prod_{i=1}^{n} f_1 \times f_2 \times f_3 \times f_4 \times f_5) \\
&= argmin \sum_{i=1}^{n} [\ln(f_1) + \ln(f_2) + \ln(f_3) + \ln(f_4) + \ln(f_5)]
\end{aligned}
$$

$$p* = argmin \sum_{i=1}^{n} \ln(f_1) + \sum_{i=1}^{n} \ln(f_2) + \sum_{i=1}^{n} \ln(f_3) + \sum_{i=1}^{n} \ln(f_4) + \sum_{i=1}^{n} \ln(f_5) \tag{1}$$

où :

$$\begin{cases} \sum_{i=1}^{n} \ln(f_1) = \sum_{i=1}^{n} [x_{i5} \ln(p1) + (1 - x_{i5}) \ln(1 - p_1)] \\ \sum_{i=1}^{n} \ln(f_2) = \sum_{i=1}^{n} [x_{i4} x_{i5} \ln(p2) + (1 - x_{i4}) x_{i5} \ln(1 - p_2) + x_{i4}(1 - x_{i5}) \ln(p3) + (1 - x_{i4})(1 - x_{i5}) \ln(1 - p3)] \\ \sum_{i=1}^{n} \ln(f_3) = \sum_{i=1}^{n} [x_{i3} x_{i4} x_{i5} \ln(p4) + (1 - x_{i3}) x_{i4} x_{i5} \ln(1 - p_4) + x_{i3}(1 - x_{i4}) x_{i5} \ln(p5) + (1 - x_{i3})(1 - x_{i4}) x_{i5} \ln(1 - p5) \\ + x_{i3} x_{i4}(1 - x_{i5}) \ln(p6) + (1 - x_{i3}) x_{i4}(1 - x_{i5}) \ln(1 - p6) + x_{i3}(1 - x_{i4})(1 - x_{i5}) \ln(p7) + (1 - x_{i3})(1 - x_{i4})(1 - x_{i5}) \\ \ln(1 - p7)] \\ \dots \end{cases}$$

Pour trouver $p*$, il faut calculer le gradient de l'équation (1) par rapport au vecteur de paramètres $p = (p_1, \dots, p_{13})$ et voir quand est-ce qu'il s'annule. Pour simplifier le calcul de toutes les dérivées partielles, soit la fonction $L' = \sum_{i=1}^{n} \ln(f_1) + \sum_{i=1}^{n} \ln(f_2) + \sum_{i=1}^{n} \ln(f_3) + \sum_{i=1}^{n} \ln(f_4) + \sum_{i=1}^{n} \ln(f_5)$ et on peut remarquer que $\forall j \in \{1 \dots 13\}$, la dérivée partielle de $L'$ par rapport au paramètre $p_j$ peut s'écrire :

$$\frac{\partial L'}{\partial p_j} = \frac{\partial(\sum_{i=1}^{n} [a_{ik} \ln(p_j) + b_{ik} \ln(1 - p_j)])}{\partial p_j} \tag{2}$$

$$= \sum_{i=1}^{n} \frac{a_{ik}}{p_j} - \sum_{i=1}^{n} \frac{b_{ik}}{1 - p_j} \tag{3}$$

où $a_{ik}$ et $b_{ik}$ sont des entiers dépendants uniquement des paramètres $(x_{i1}, \dots, x_{i5})$ provenant de la fonction $f_k$ et d'où $p_j$ est aussi l'un des paramètres. En effet, nous arrivons à cette expression car $L'$ s'écrit comme une somme de différentes sommes de fonctions $\ln(f_k)$ pour $\forall k \in \{1 \dots 5\}$. Bien plus, comme nous l'avons vu, $\forall k \in \{1 \dots 5\}$ la somme suivante $\sum_{i=1}^{n} \ln(f_k)$ s'exprime aussi comme une somme de différentes expressions $\sum_{i=1}^{n} [a_{ik} \ln(p_j) + b_{ik} \ln(1 - p_j)]$ où les $p_j$ sont les paramètres de la fonction $f_k$ et $a_{ik}$, $b_{ik}$ sont des variables dépendantes uniquements des paramètres $(x_{i1}, \dots, x_{i5})$ de $f_k$. Par conséquent, en prenant en compte les règles de dérivation de la somme, la dérivée partielle de $L'$ par rapport au paramètre $p_j$ est de la forme de l'équation (3). Essayons maintenant de montrer quand est-ce l'équation (3) s'annule :

$$\begin{aligned} \sum_{i=1}^{n} \frac{a_{ik}}{p_j} - \sum_{i=1}^{n} \frac{b_{ik}}{1 - p_j} &= 0 \\ \sum_{i=1}^{n} \frac{a_{ik}}{p_j} &= \sum_{i=1}^{n} \frac{b_{ik}}{1 - p_j} \\ \frac{1 - p_j}{p_j} \sum_{i=1}^{n} a_{ik} &= \sum_{i=1}^{n} b_{ik} \\ \frac{1 - p_j}{p_j} &= \frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}} \\ \frac{1}{p_j} &= \frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}} + 1 \\ p_j &= \frac{\sum_{i=1}^{n} a_{ik}}{\sum_{i=1}^{n} a_{ik} + \sum_{i=1}^{n} b_{ik}} \end{aligned}$$

Pour achever la démonstration, il faut montrer que les points trouvés $(p_1, \dots, p_{13})$ donnent bien le minimum de la fonction $L(X_i, p)$. Pour cela, on peut calculer la hessienne de la fonction $L(X_i, p)$ et montrer qu'elle est définie positive aux points $p_j$ trouvés. Tout d'abord, calculons la dérivée partielle seconde par rapport à $p_j$ :

$$\frac{\partial^2 L'}{\partial p_j \, \partial p_j} = -\sum_{i=1}^{n} \frac{a_{ik}}{p_j^2} - \sum_{i=1}^{n} \frac{b_{ik}}{(1-p_j)^2} \tag{4}$$

Notons tout d'abord que $\forall p_j \in \{p_1, \ldots, p_{13}\}$ on a $\frac{\partial^2 L'}{\partial p_j \, \partial p_j} < 0$ (ce qui est évident). Maintenant, montrons que l'équation (4) est strictement supérieure à 0 en tout point $p_j = \frac{\sum_{i=1}^{n} a_{ik}}{\sum_{i=1}^{n} a_{ik} + \sum_{i=1}^{n} b_{ik}}$ :

$$\begin{aligned}
-\sum_{i=1}^{n} \frac{a_{ik}}{p_j^2} - \sum_{i=1}^{n} \frac{b_{ik}}{(1-p_j)^2} &< 0 \\
-\sum_{i=1}^{n} \frac{a_{ik}}{p_j^2} &< \sum_{i=1}^{n} \frac{b_{ik}}{(1-p_j)^2} \\
-\frac{(1-p_j)^2}{p_j^2} &< \frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}} \\
\frac{-p_j^2 + 2p_j - 1}{p_j^2} &< \frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}} \\
-p_j + 2 - \frac{1}{p_j} &< p_j \times \frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}} \\
-2 \times \frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}} &< (\frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}})^2 + \frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}} \\
(\frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}})^2 + 3 \times \frac{\sum_{i=1}^{n} b_{ik}}{\sum_{i=1}^{n} a_{ik}} &> 0
\end{aligned}$$

Nous pouvons maintenant conclure : en effet, la matrice hessienne de la fonction $L(X_i, p)$ est de la forme suivante :

$$H(L(X_i,p))(p_1 \ldots p_{13}) = \begin{bmatrix} \frac{\partial^2 L'}{\partial p_1 \, \partial p_1}(p_1) & 0 & \ldots & 0 \\ 0 & \frac{\partial^2 L'}{\partial p_2 \, \partial p_2}(p_2) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{\partial^2 L'}{\partial p_{13} \, \partial p_{13}}(p_{13}) \end{bmatrix}$$

$H(L(X_i, p))(p_1 \ldots p_{13})$ est une matrice diagonale et ses valeurs propres sont donc ceux de sa diagonale. Or, sa diagonale contient que des valeurs strictement positives car on a démontré que pour $\forall p_j \in \{p_1, \ldots, p_{13}\}$ on a $\frac{\partial^2 L'}{\partial p_j \, \partial p_j}(pj) > 0$. Ainsi, ses valeurs propres sont aussi strictement positives et $H(L(X_i, p))(p_1 \ldots p_{13})$ est donc définie positivement. Par conséquent, les points trouvés $(p_1, \ldots, p_{13})$ à l'aide de la formule $p_j = \frac{\sum_{i=1}^{n} a_{ik}}{\sum_{i=1}^{n} a_{ik} + \sum_{i=1}^{n} b_{ik}}$ fournissent bien le minimum de la fonction $L(X_i, p)$.

Pour estimer le vecteur de paramètres $p = (p_1, \ldots, p_{13})$ à l'aide de la matrice *notesReussite*, nous allons passer par deux étapes :

- la première c'est de déterminer pour chaque $p_j$ ses sommes $\sum_{i=1}^{n} a_{ik}$ et $\sum_{i=1}^{n} b_{ik}$ à l'aide de la matrice *notesReussite*.
- lorsqu'on a obtenu la valeur de ces deux sommes, on peut appliquer la formule trouvée auparavant à l'aide du maximum de vraisemblance (i.e $p_j = \frac{\sum_{i=1}^{n} a_{ik}}{\sum_{i=1}^{n} a_{ik} + \sum_{i=1}^{n} b_{ik}}$) afin d'estimer au mieux $p_j$.

Voici maintenant l'implémentation de la solution analytique sous R :

```
maxVraisemblance = function(notesReussite)
{
  n = nrow(notesReussite)
  res = matrix(0, nrow = 13, ncol = 2)

  # compute sum(aik) and sum(bik) for each pj
  for(i in 1:n)
  {
    Xi = unlist(notesReussite[i,], use.names = FALSE)

    res[1,1] = res[1,1] + Xi[5]
    res[1,2] = res[1,2] + (1-Xi[5])

    res[2,1] = res[2,1] + Xi[4]*Xi[5]
    res[2,2] = res[2,2] + (1-Xi[4])*Xi[5]
    res[3,1] = res[3,1] + Xi[4]*(1-Xi[5])
    res[3,2] = res[3,2] + (1-Xi[4])*(1-Xi[5])

    res[4,1] = res[4,1] + Xi[3]*Xi[4]*Xi[5]
    res[4,2] = res[4,2] + (1-Xi[3])*Xi[4]*Xi[5]
    res[5,1] = res[5,1] + Xi[3]*(1-Xi[4])*Xi[5]
    res[5,2] = res[5,2] + (1-Xi[3])*(1-Xi[4])*Xi[5]
    res[6,1] = res[6,1] + Xi[3]*Xi[4]*(1-Xi[5])
    res[6,2] = res[6,2] + (1-Xi[3])*Xi[4]*(1-Xi[5])
    res[7,1] = res[7,1] + Xi[3]*(1-Xi[4])*(1-Xi[5])
    res[7,2] = res[7,2] + (1-Xi[3])*(1-Xi[4])*(1-Xi[5])

    res[8,1] = res[8,1] + Xi[2]*Xi[3]
    res[8,2] = res[8,2] + (1-Xi[2])*Xi[3]
    res[9,1] = res[9,1] + Xi[2]*(1-Xi[3])
    res[9,2] = res[9,2] + (1-Xi[2])*(1-Xi[3])

    res[10,1] = res[10,1] + Xi[1]*Xi[2]*Xi[3]
    res[10,2] = res[10,2] + (1-Xi[1])*Xi[2]*Xi[3]
    res[11,1] = res[11,1] + Xi[1]*(1-Xi[2])*Xi[3]
    res[11,2] = res[11,2] + (1-Xi[1])*(1-Xi[2])*Xi[3]
    res[12,1] = res[12,1] + Xi[1]*Xi[2]*(1-Xi[3])
    res[12,2] = res[12,2] + (1-Xi[1])*Xi[2]*(1-Xi[3])
    res[13,1] = res[13,1] + Xi[1]*(1-Xi[2])*(1-Xi[3])
    res[13,2] = res[13,2] + (1-Xi[1])*(1-Xi[2])*(1-Xi[3])
  }

  p = c(rep(0,13))

  # compute each pj
  for(i in 1:13)
  {
    p[i] = res[i,1] / (res[i,1] + res[i,2])
  }

  return(p)
}
```

```r
# use the int version of notesReussite instead
notesReussiteInt = as.data.frame((marks >= 45) * 1)
p = maxVraisemblance(notesReussiteInt)
print(p)
```

```
##  [1] 0.3863636 0.8529412 0.5555556 0.9655172 0.8000000 0.8333333 0.4166667
##  [8] 0.7761194 0.2857143 0.5576923 0.3333333 0.3333333 0.1333333
```

Par conséquent :

$$\begin{cases} p_1 = \mathbb{P}(X_{i5} = 1) = 0.3863636 \\ p_2 = \mathbb{P}(X_{i4} = 1 | X_{i5} = 1) = 0.8529412 \\ p_3 = \mathbb{P}(X_{i4} = 1 | X_{i5} = 0) = 0.5555556 \\ p_4 = \mathbb{P}(X_{i3} = 1 | X_{i4} = 1, X_{i5} = 1) = 0.9655172 \\ p_5 = \mathbb{P}(X_{i3} = 1 | X_{i4} = 0, X_{i5} = 1) = 0.8000000 \\ p_6 = \mathbb{P}(X_{i3} = 1 | X_{i4} = 1, X_{i5} = 0) = 0.8333333 \\ p_7 = \mathbb{P}(X_{i3} = 1 | X_{i4} = 0, X_{i5} = 0) = 0.4166667 \\ p_8 = \mathbb{P}(X_{i2} = 1 | X_{i3} = 1) = 0.7761194 \\ p_9 = \mathbb{P}(X_{i2} = 1 | X_{i3} = 0) = 0.2857143 \\ p_{10} = \mathbb{P}(X_{i1} = 1 | X_{i2} = 1, X_{i3} = 1) = 0.5576923 \\ p_{11} = \mathbb{P}(X_{i1} = 1 | X_{i2} = 0, X_{i3} = 1) = 0.3333333 \\ p_{12} = \mathbb{P}(X_{i1} = 1 | X_{i2} = 1, X_{i3} = 0) = 0.3333333 \\ p_{13} = \mathbb{P}(X_{i1} = 1 | X_{i2} = 0, X_{i3} = 0) = 0.1333333 \end{cases}$$

## 4.

```r
bn.fit(dagNotes, data = notesReussite)
```

```
##
##   Bayesian network parameters
##
##   Parameters of node MECH (multinomial distribution)
##
## Conditional probability table:
##
## , , ALG = E
##
##     VECT
## MECH         E         R
##    E 0.8666667 0.6666667
##    R 0.1333333 0.3333333
##
## , , ALG = R
##
##     VECT
## MECH         E         R
##    E 0.6666667 0.4423077
##    R 0.3333333 0.5576923
##
##   Parameters of node VECT (multinomial distribution)
##
```

p12

p13

p10

p11

```
## Conditional probability table:
##
##     ALG
## VECT         E         R
##    E 0.7142857 0.2238806
##    R 0.2857143 0.7761194 p8
##         p9
##   Parameters of node ALG (multinomial distribution)
##
## Conditional probability table:
##
## , , STAT = E
##
##    ANL
## ALG          E          R
##   E 0.58333333 0.16666667
##   R 0.41666667 0.83333333 p6
##        p7
## , , STAT = R
##
##    ANL
## ALG          E          R
##   E 0.20000000 0.03448276
##   R 0.80000000 0.96551724 p4
##        p5
##
##   Parameters of node ANL (multinomial distribution)
##
## Conditional probability table:
##
##    STAT
## ANL         E         R
##   E 0.4444444 0.1470588
##   R 0.5555556 0.8529412 p2
##      p3
##   Parameters of node STAT (multinomial distribution)
##
## Conditional probability table:
##          E         R
## 0.6136364 0.3863636  p1
```

On remarque que nous obtenons les mêmes résultats qu'à la section 4) (voir les annotations au niveau de l'output de la faction *bn.fit*).