STT760 : MATHEMATIQUE POUR L'INTELLIGENCE ARTIFICIELLE

Yves Lolelo
Noudéhouénou Houessou
Laetitia Meuleghe Kenmegne
Oussama Khaloui
Gbogou Henri-Michel

2023-2024

STT760 Mathématique de l'Intelligence Artificielle

2023-10-25

Installation des packages

```
if (!requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager")
BiocManager::install(c("gRain", "gRbase","graph","RBGL","Rgraphviz", "ggm"))
```

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   `force = TRUE` to re-install: 'gRain' 'gRbase' 'graph' 'RBGL' 'Rgraphviz'
##   'ggm'
```

```
library("gRain")
library("gRbase")
library("Rgraphviz")
library("ggm")
```

1 - Représentation graphique du réseau Bayésien

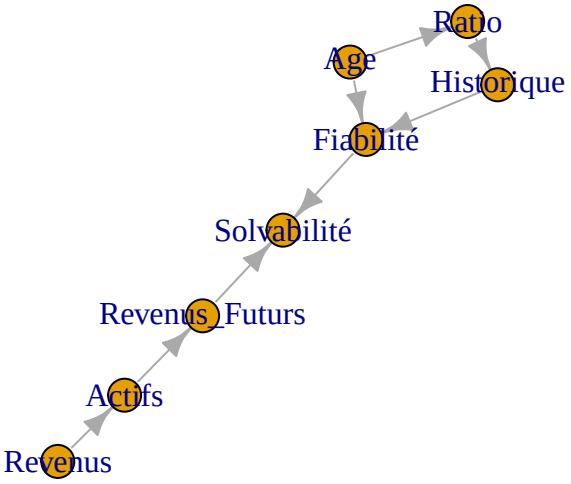
Création d'un objet GraphNel

```
dag_q1 <- dag(~Fiabilité|Historique:Age, ~Ratio|Age, ~Historique|Ratio,
              ~Actifs|Revenus, ~Revenus_Futurs|Actifs,
              ~Solvabilité|Fiabilité:Revenus_Futurs)

dag_q1
```

```
## IGRAPH d1b9053 DN-- 8 8 --
## + attr: name (v/c)
## + edges from d1b9053 (vertex names):
## [1] Historique    ->Fiabilité      Age           ->Fiabilité
## [3] Age           ->Ratio          Ratio         ->Historique
## [5] Revenus       ->Actifs         Actifs        ->Revenus_Futurs
## [7] Fiabilité     ->Solvabilité    Revenus_Futurs->Solvabilité
```

Visualisation du graphe orienté $\mathcal{G}$

```
plot(dag_q1, ,edge.arrow.width=1,edge.arrow.size=0.8)
```



Claude croit que la *solvabilité* et le *ratio dettes vs revenus* sont liés; i.e qu'il y a une dépendance entre la *solvabilité* et le *ratio dettes vs revenus*. A la lecture du graphe $\mathcal{G}$, étant donné que la *fiabilité* est liée à la *solvabilité*, il faudra juste s'assurer que le *ratio dettes vs revenus* et la *fiabilité* sont dépendants. Pour ce faire, déterminons la matrice d'adjacence de $\mathcal{G}$ et vérifierons que les deux noeuds ne sont pas indépendants via le critère de la d-séparation.

Matrice d'adjacence de $\mathcal{G}$

```
Adj_dag_q1 = as(dag_q1,"matrix")
Adj_dag_q1
```

```
##                Historique Fiabilité Age Ratio Revenus Actifs Revenus_Futurs
## Historique              0         1   0     0       0      0              0
## Fiabilité               0         0   0     0       0      0              0
## Age                     0         1   0     1       0      0              0
## Ratio                   1         0   0     0       0      0              0
## Revenus                 0         0   0     0       0      1              0
## Actifs                  0         0   0     0       0      0              1
## Revenus_Futurs          0         0   0     0       0      0              0
## Solvabilité             0         0   0     0       0      0              0
##                Solvabilité
## Historique               0
## Fiabilité                1
## Age                      0
## Ratio                    0
## Revenus                  0
## Actifs                   0
## Revenus_Futurs           1
## Solvabilité              0
```

Vérifions la dépendence entre le ratio et la fiabilité

```
dSep(Adj_dag_q1,"Fiabilité","Ratio", "Age")
```

```
## [1] FALSE
```

```
dSep(Adj_dag_q1,"Fiabilité","Ratio", "Historique")
```

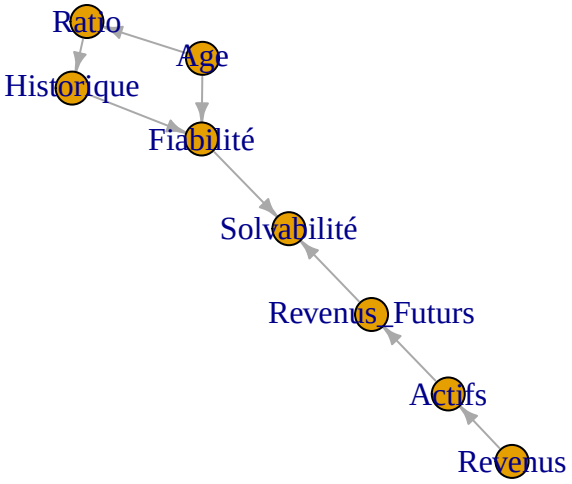
```
## [1] FALSE
```

2- Elicitation des probabilités

```
val = c("Élevé", "Moyen", "Faible") # valeurs possibles pour les variables...
                                    # ..Revenus, Actifs, Revenus_Futurs, Ratio

cp_Revenus <- cptable(~Revenus, levels=val,
                      values=c(15,55,30))

cp_Actifs <- cptable(~Actifs|Revenus, levels=val,
                     values=c(80,15,5,30,50,20,10,30,60))

cp_Revenus_Futurs <- cptable(~Revenus_Futurs|Actifs, levels=val,
                             values=c(80,15,5,30,50,20,10,30,60))

cp_Solvabilité <- cptable(~Solvabilité|Revenus_Futurs+Fiabilité,
                          levels=c("Solvable", "Non Solvable"),
                          values=c(85,15,75,25,70,30,55,45,65,35,60,40))

cp_Fiabilité <- cptable(~Fiabilité|Historique+Age,
                        levels=c("Fiable", "Non Fiable"),
                        values=c(50,50,45,55,60,40,55,45,70,30,65,35,90,10,80,20))


cp_Historique <- cptable(~Historique|Ratio, levels=c("Bon", "Mauvais"),
                         values=c(55,45,75,25,90,10))

cp_Age <- cptable(~Age, levels=c("<25", "25-50", "50-65", ">65"),
                  values=c(25, 45, 20, 10))

cp_Ratio <- cptable(~Ratio|Age, levels=val,
                    values=c(45,30,25,35,25,40,10,15,75,5,10,85))
```

Création d'un objet de type grain (GRAphical Independence Network) de la librairie *gRain*

```
net_list = compileCPT(list(cp_Revenus, cp_Actifs,cp_Revenus_Futurs,
                           cp_Solvabilité, cp_Fiabilité,cp_Historique,
                           cp_Age, cp_Ratio))

grain_q2 <- grain(net_list)

plot(grain_q2$dag,
     edge.arrow.width=1,
     edge.arrow.size=0.5)
```



Calcul des probabilités et vérification des 8 contraintes

- *Condition 1 : Un(e) client(e) avec un bon historique de paiement a tendance à être plus fiable*

```
p1 <- querygrain(grain_q2, nodes = c("Fiabilité","Historique"), type = "conditional")
p1[c("Fiable"), c("Bon")] > p1[c("Fiable"), c("Mauvais")]
```

```
## [1] TRUE
```

- *Condition 2 : Plus un(e) client(e) est âgé(e), plus il/elle a de chance d'être fiable*

```
p2 <- querygrain(grain_q2, nodes = c("Fiabilité","Age"), type = "conditional")
p2[c("Fiable"), c("25-50")] > p2[c("Fiable"), c("<25")] &
  p2[c("Fiable"), c("50-65")] > p2[c("Fiable"), c("25-50")] &
  p2[c("Fiable"), c(">65")] > p2[c("Fiable"), c("50-65")]
```

```
## [1] TRUE
```

- *Condition 3 : Les clients plus âgés ont tendance à avoir un fiable ratio dettes vs revenus*

```
p3 <- querygrain(grain_q2, nodes = c("Ratio","Age"), type = "conditional")
p3[c("Faible"), c("25-50")] > p3[c("Faible"), c("<25")] &
  p3[c("Faible"), c("50-65")] > p3[c("Faible"), c("25-50")] &
  p3[c("Faible"), c(">65")] > p3[c("Faible"), c("50-65")]
```

```
## [1] TRUE
```

- *Condition 4 : La probabilité d'avoir un bon historique de paiement augmente au fur et à mesure que le ratio de dette vs revenus diminue*

```
p4 <- querygrain(grain_q2, nodes = c("Historique","Ratio"), type = "conditional")
p4[c("Bon"), c("Moyen")] > p4[c("Bon"), c("Élevé")] &
  p4[c("Bon"), c("Faible")] > p4[c("Bon"), c("Moyen")]
```

```
## [1] TRUE
```

- *Condition 5 : Plus les revenus d'une personne sont élevés, plus cette personne a de chance d'avoir des actifs élevés*

```
p5 <- querygrain(grain_q2, nodes = c("Actifs","Revenus"), type = "conditional")
p5[c("Élevé"), c("Moyen")] > p5[c("Élevé"), c("Faible")] &
  p5[c("Élevé"), c("Élevé")] > p5[c("Élevé"), c("Moyen")]
```

```
## [1] TRUE
```

- *Condition 6 : Plus une personne a d'actifs, plus cette
personne a de chance d'avoir un revenu élevé dans le futur*

```
p6 <- querygrain(grain_q2, nodes = c("Revenus_Futurs","Actifs"), type = "conditional")
p6[c("Élevé"), c("Moyen")] > p6[c("Élevé"), c("Faible")] &
  p6[c("Élevé"), c("Élevé")] > p6[c("Élevé"), c("Moyen")]
```

```
## [1] TRUE
```

- *Condition 7 : Une personne fiable a tendance à être plus solvable qu'une personne non fiable*

```
p7 <- querygrain(grain_q2, nodes = c("Solvabilité","Fiabilité"), type = "conditional")
p7[c("Solvable"), c("Fiable")] > p7[c("Solvable"), c("Non Fiable")]
```

```
## [1] TRUE
```

- *Condition 8 : Les personnes qui ont des revenus prometteurs
ont plus de chance d'être solvables que celles dont la perspective
des revenus à venir est mauvaise*

```
p8 <- querygrain(grain_q2, nodes = c("Solvabilité","Revenus_Futurs"), type = "conditional")
p8[c("Solvable"), c("Moyen")] > p8[c("Solvable"), c("Faible")] &
  p8[c("Solvable"), c("Élevé")] > p8[c("Solvable"), c("Moyen")]
```

```
## [1] TRUE
```

STT760 MATHEMATIQUE POUR l'INTELLIGENCE ARTIFICIELLE

2023-10-25

```
if (!requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager")
BiocManager::install(c("gRain", "gRbase","graph","RBGL","Rgraphviz"))
```

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'gRain' 'gRbase' 'graph' 'RBGL' 'Rgraphviz'
```

```
library("gRain")
library("gRbase")
library("Rgraphviz")
```

1 - Prépartion des données

```
# install packages ("bnlearn")
library("bnlearn")
```

```
##
## Attaching package: 'bnlearn'
```

```
## The following objects are masked from 'package:gRbase':
##
##     ancestors, children, parents
```

```
data(marks)
names(marks)
```

```
## [1] "MECH" "VECT" "ALG"  "ANL"  "STAT"
```

```
summary(marks)
```

```
##       MECH            VECT            ALG             ANL
##  Min.   : 0.00   Min.   : 9.00   Min.   :15.00   Min.   : 9.00
##  1st Qu.:30.00   1st Qu.:42.00   1st Qu.:45.00   1st Qu.:35.75
##  Median :41.50   Median :51.00   Median :50.00   Median :49.00
##  Mean   :38.95   Mean   :50.59   Mean   :50.60   Mean   :46.68
##  3rd Qu.:49.25   3rd Qu.:60.00   3rd Qu.:57.25   3rd Qu.:57.00
```

```
##  Max.   :77.00   Max.   :82.00   Max.   :80.00   Max.   :70.00
##       STAT
##  Min.   : 9.00
##  1st Qu.:31.00
##  Median :40.00
##  Mean   :42.31
##  3rd Qu.:51.50
##  Max.   :81.00
```

```
notes_reussite = (marks >=45)*1
#notes_reussite[notes_reussite==1] = "R"
#notes_reussite[notes_reussite==0] = "E"
```

Dans la matrice **notes_reussite**, un **R** indique une réussite et un **E** indique un échec.

2- Fonctions pour paramétriser $\mathbb{P}_{Xi}$

$X_i = (X_{i1}, ..., X_{i5})$ est un vecteur aléatoire tel que X_{i1} (respectivement $(X_{i2}..., X_{i5})$) vaut 1 si l'étudiant(e) i a réussi mécanique (resp. algèbre vectorielle, algèbre, analyse et statistique)

a) Elicitons les fonctions f_1, f_2, f_3, f_4 qui permettent de paramétriser $\mathbb{P}_{Xi}$

```
f1 <- function(x,p1){
  e=0
  if((x==0) || (x==1))
  {
    e = (p1^x) * (1-p1)^(1-x)
  }
  return(e)
}
```

```
f2 <- function(x,y,p2,p3){
  e=0
  if((x==0) || (x==1) & ((y==0) || (y==1)))
  {
    e = ((p2^x * (1-p2)^(1-x))^y) * ((p3^x * (1-p3)^(1-x))^(1-y))
  }
  return(e)
}
```

```
f3 <- function(x,y,z,p4,p5,p6,p7) {
   e=0
   f=0
   g=0
   h=0
   t=0
  if((x==0) || (x==1) & ((y==0) || (y==1)) & ((z==0) || (z==1)))
  {
   e = p4^x * (1-p4)^(1-x)
   f = p5^x * (1-p5)^(1-x)
   g = p6^x * (1-p6)^(1-x)
```

```
   h = p7^x * (1-p7)^(1-x)
   t = (e^(y*z) * f^(z*(1-y))* g^(y*(1-z))* h^((1-y)*(1-z)))
  }
  return (t)
}
```

```
f4 <- function(x,y,p8,p9){
  e=0
  if((x==0) || (x==1) & ((y==0) || (y==1)))
  {
    e = ((p8^x * (1-p8)^(1-x))^y) * ((p9^x * (1-p9)^(1-x))^(1-y))
  }
  return (e)
}
```

```
f5 <- function(x,y,z,p10,p11,p12,p13) {
   e=0
   f=0
   g=0
   h=0
   t=0
  if((x==0) || (x==1) & ((y==0) || (y==1)) & ((z==0) || (z==1)))
  {
   e = p10^x * (1-p10)^(1-x)
   f = p11^x * (1-p11)^(1-x)
   g = p12^x * (1-p12)^(1-x)
   h = p13^x * (1-p13)^(1-x)
   t = (e^(y*z)*f^(z*(1-y))*g^(y*(1-z))*h^((1-y)*(1-z)))
  }
  return (t)
}
```

b) La fonction de vraisemblance se présente comme suit :

```
L_p <- function(x,y,z,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13){

  return(f1(x,p1)*f2(x,y,p2,p3)*f3(x,y,z,p4,p5,p6,p7)*f4(x,y,p8,p9)*
           f5(x,y,z,p10,p11,p12,p13))
}
```

3 - Ajustement de paramètres par maximum de vraisemblance

Formellement, étant donné un échantillon d'observations indépendantes et de même distribution, on a :

$$p^* = argmax \prod_{i=1}^{n} L(x_i, p)$$

- De la question précédente, la fonction de vraisemblance se présente comme suit :

```
L_p <- function(x,y,z,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13){

  return(f1(x,p1)*f2(x,y,p2,p3)*f3(x,y,z,p4,p5,p6,p7)*f4(x,y,p8,p9)*
           f5(x,y,z,p10,p11,p12,p13))
}
```

- Les différentes valeurs de p^* obtenues analytiquement se présentent comme suit :

```
Phat_1 <- function(X5) {return(sum(X5)/length(X5))}

Phat_2 <- function(X4,X5) {return(sum(X4*X5)/sum(X5))}

Phat_3 <- function(X4,X5) { return(sum((1-X5)*X4)/sum(1-X5))}

Phat_4 <- function(X3, X4, X5) { return(sum(X3*X4*X5)/sum(X4*X5))}

Phat_5 <- function(X3, X4, X5) {return(sum(X3*X4*(1-X5))/sum(X4*(1-X5)))}

Phat_6 <- function(X3, X4, X5) { return(sum(X3*X5*(1-X4))/sum((1-X4)*X5))}

Phat_7 <- function(X3, X4, X5) {return(sum(X3*(1-X4)*(1-X5))/sum((1-X4)*(1-X5)))}

Phat_8 <- function(X2,X3) {return(sum(X2*X3)/sum(X3))}

Phat_9 <- function(X2,X3) {return(sum(X2*(1-X3))/sum(1-X3))}

Phat_10 <- function(X1, X2, X3) {return(sum(X1*X2*X3)/sum(X2*X3))}

Phat_11 <- function(X1, X2, X3) {return(sum(X1*X2*(1-X3))/sum(X2*(1-X3)))}

Phat_12 <- function(X1, X2, X3) {return(sum(X1*(1-X2)*X3)/sum(X3*(1-X2)))}

Phat_13 <- function(X1, X2, X3) {return(sum(X1*(1-X2)*(1-X3))/sum((1-X2)*(1-X3)))}
```

- Calculons la valeur de p^*

```
notes_reussite = (marks >=45)*1
X1 <- notes_reussite[,1]
X2 <- notes_reussite[,2]
X3 <- notes_reussite[,3]
X4 <- notes_reussite[,4]
X5 <- notes_reussite[,5]

Phat <- c(Phat_1(X5), Phat_2(X4, X5), Phat_3(X4, X5), Phat_4(X3, X4, X5),
          Phat_5(X3,X4,X5), Phat_6(X3,X4,X5), Phat_7(X3,X4,X5), Phat_8(X2, X3),
          Phat_9(X2, X3), Phat_10(X1,X2,X3), Phat_11(X1,X2,X3), Phat_12(X1,X2,X3), Phat_13(X1,X2,X3))

Phat
```

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

Ainsi, on a:

$$\mathbb{P}_1 = \mathbb{P}(STAT = 1) = 0.3863636$$

$$\mathbb{P}_2 = \mathbb{P}(ANL = 1|STAT = 1) = 0.8529412$$

$$\mathbb{P}_3 = \mathbb{P}(ANL = 1|STAT = 0) = 0.5555556$$

$$\mathbb{P}_4 = \mathbb{P}(ALG = 1|ANL = 1, STAT = 1) = 0.9655172$$

$$\mathbb{P}_5 = \mathbb{P}(ALG = 1|ANL = 1, STAT = 0) = 0.8333333$$

$$\mathbb{P}_6 = \mathbb{P}(ALG = 1|ANL = 0, STAT = 1) = 0.8000000$$

$$\mathbb{P}_7 = \mathbb{P}(ALG = 1|ANL = 0, STAT = 0) = 0.4166667$$

$$\mathbb{P}_8 = \mathbb{P}(VECT = 1|ALG = 1) = 0.7761194$$

$$\mathbb{P}_9 = \mathbb{P}(VECT = 1|ALG = 0) = 0.2857143$$

$$\mathbb{P}_{10} = \mathbb{P}(MECH = 1|VECT = 1, ALG = 1) = 0.5576923$$

$$\mathbb{P}_{11} = \mathbb{P}(MECH = 1|VECT = 1, ALG = 0) = 0.3333333$$

$$\mathbb{P}_{12} = \mathbb{P}(MECH = 1|VECT = 0, ALG = 1) = 0.3333333$$

$$\mathbb{P}_{13} = \mathbb{P}(MECH = 1|VECT = 0, ALG = 0) = 0.1333333$$

4- Comparons la solution trouvée avec le résultat de la fonction *bn.fit*

La fonction *bn.fit* est contenue dans la librairie **bnlearn** qui représente un réseau bayésien par l'entremise d'un objet de type *bn*. Pour ce faire :

- (a) Construisons un graphe $\mathcal{G}_\infty$ sans arête à partir d'une liste de noeuds

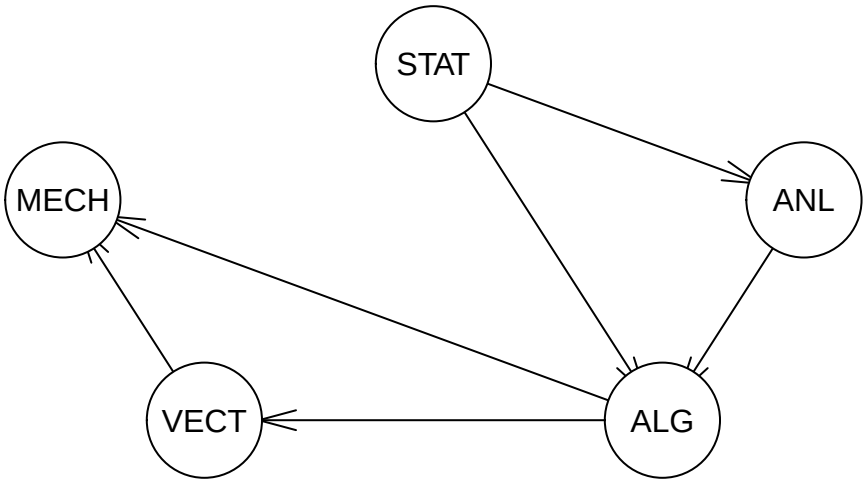
```
graphe_notes = empty.graph(names(marks))
```

- (b) Modifions un graphe $\mathcal{G}_\infty$ en y ajoutant des arêtes

```
arcs(graphe_notes) = matrix(
c("VECT", "MECH", "ALG", "MECH", "ALG", "VECT",
  "ANL", "ALG", "STAT", "ALG","STAT", "ANL"),
   ncol = 2, byrow = TRUE, dimnames = list(c(), c("from", "to")))
graphe_notes
```

```
##
##   Random/Generated Bayesian network
##
##   model:
##    [STAT][ANL|STAT][ALG|ANL:STAT][VECT|ALG][MECH|VECT:ALG]
##   nodes:                                 5
##   arcs:                                  6
##     undirected arcs:                     0
##     directed arcs:                       6
##   average markov blanket size:           2.40
##   average neighbourhood size:            2.40
##   average branching factor:              1.20
##
##   generation algorithm:                  Empty
```

```
plot(graphe_notes)
```



- (c) Spécifions les paires d'arêtes en élicitant la matrice d'adjacence associée au graphe $\mathcal{G}_\infty$ comme c'est le cas pour les objets *GraphNEL*

```
matrice_ad = matrix(c(0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
                      0, 0, 1, 0, 0, 0, 0, 0), nrow = 5,
                    dimnames = list(names(marks), names(marks)))
matrice_ad
```

```
##      MECH VECT ALG ANL STAT
## MECH    0    0   0   0    0
## VECT    1    0   0   0    0
## ALG     1    1   0   0    0
## ANL     0    0   1   0    0
## STAT    0    0   1   1    0
```

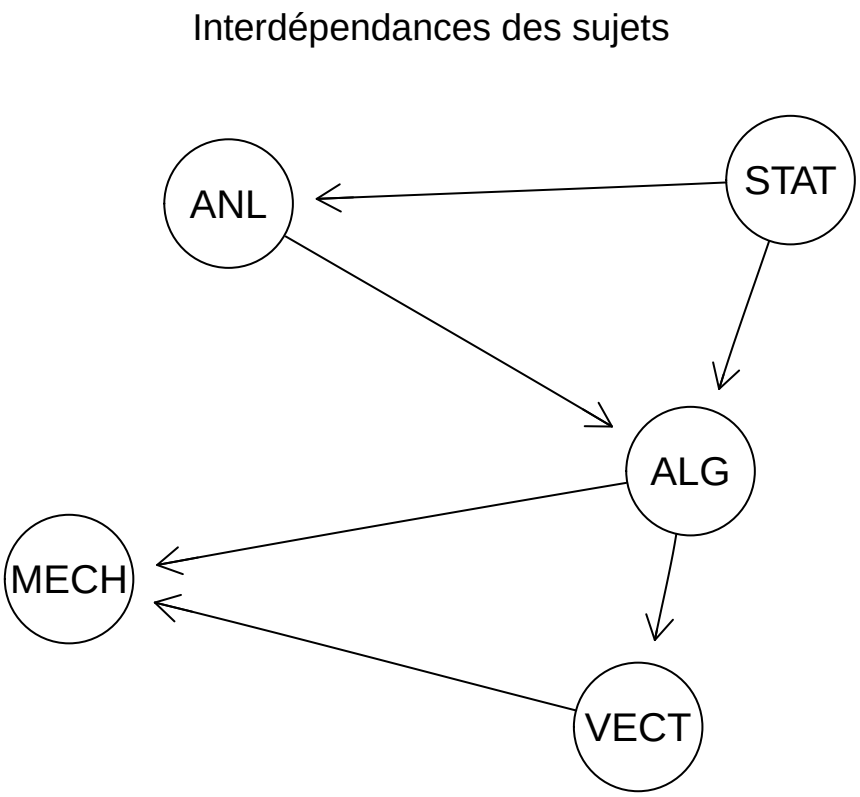
Ainsi, la strucutre du réseau est créée.

```
graphe2_notes = empty.graph(names(marks))
amat(graphe2_notes) = matrice_ad
all.equal(graphe_notes, graphe2_notes)
```

```
## [1] TRUE
```

- (d) Visualisons la structure de réseau Bayésien

```r
v_stb = list(arcs = vstructs(graphe2_notes, arcs = TRUE), lwd = 4, col = "black")
graphviz.plot(graphe2_notes, highlight = v_stb, layout = "fdp", main = "Interdépendances des sujets")
```



- (e) Vérifions les paramètres du modèle

La fonction *bn.fit* appliquée directement suppose par défaut que nos variables suivent ne distribution gaussienne. Dans le cas présent, nos variables suivent une loi binomiale. Pour y palier :

- Discrétisation de la base de données

```r
data1<-as.data.frame(notes_reussite)
data1[,'MECH']<-factor(data1[,'MECH'])
data1[,'ALG']<-factor(data1[,'ALG'])
data1[,'VECT']<-factor(data1[,'VECT'])
data1[,'STAT']<-factor(data1[,'STAT'])
data1[,'ANL']<-factor(data1[,'ANL'])
```

- Elicitons les probabilités

```r
bn.fit(graphe2_notes, data = data1)
```

```
##
##   Bayesian network parameters
```

```
##
##   Parameters of node MECH (multinomial distribution)
##
## Conditional probability table:
##
## , , ALG = 0
##
##     VECT
## MECH         0         1
##    0 0.8666667 0.6666667
##    1 0.1333333 0.3333333
##
## , , ALG = 1
##
##     VECT
## MECH         0         1
##    0 0.6666667 0.4423077
##    1 0.3333333 0.5576923
##
##
##   Parameters of node VECT (multinomial distribution)
##
## Conditional probability table:
##
##     ALG
## VECT         0         1
##    0 0.7142857 0.2238806
##    1 0.2857143 0.7761194
##
##   Parameters of node ALG (multinomial distribution)
##
## Conditional probability table:
##
## , , STAT = 0
##
##    ANL
## ALG          0          1
##   0 0.58333333 0.16666667
##   1 0.41666667 0.83333333
##
## , , STAT = 1
##
##    ANL
## ALG          0          1
##   0 0.20000000 0.03448276
##   1 0.80000000 0.96551724
##
##
##   Parameters of node ANL (multinomial distribution)
##
## Conditional probability table:
##
##    STAT
## ANL         0         1
```

```
##   0 0.4444444 0.1470588
##   1 0.5555556 0.8529412
##
##   Parameters of node STAT (multinomial distribution)
##
## Conditional probability table:
##          0         1
## 0.6136364 0.3863636
```

En comparant avec les probabilités obtenues à la question 3, on conclut que les résultats sont identiques. La fonction bn.fit* nous donne donc les mêmes estimateurs que ceux du MLE*