Exercice_R

Yunfan CAI, Dylan SI, Victor Chau, Pierre Bonneau

2022-10-28

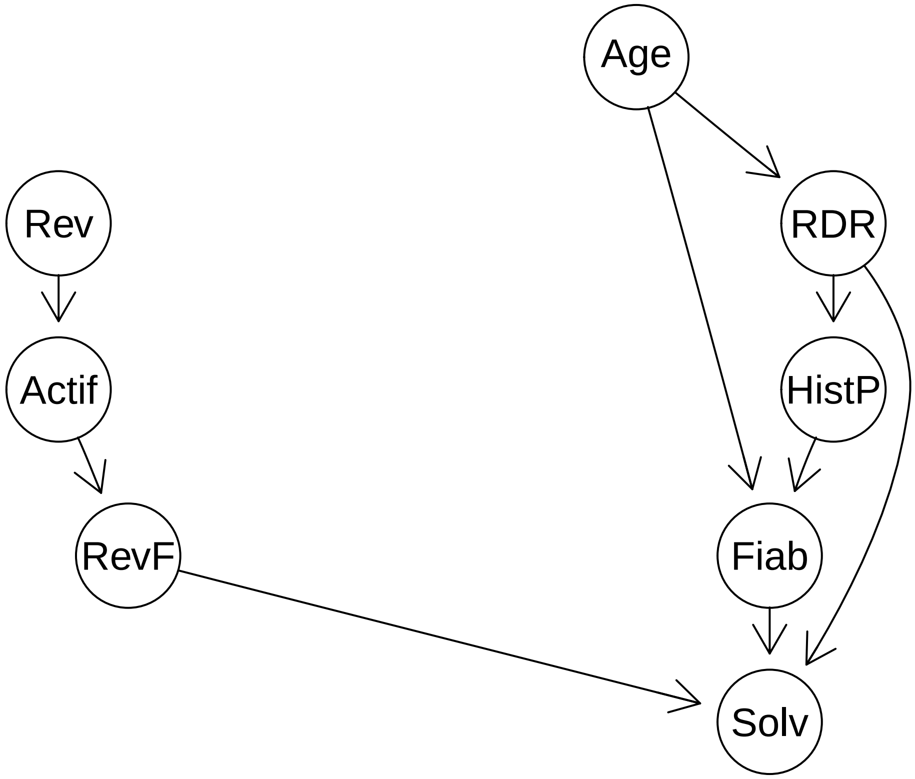
EXERCICE 1

Claude, qui travaille dans une banque, a appris votre nouvelle expertise dans la modélisation de systèmes par réseaux bayésiens, et demande votre aide pour construire un modèle prédictif de la solvabilité de ses client(e)s. Claude vous mentionne que les variables suivantes sont accessibles pour la construction du modèle prédictif pour chaque client(e): • Revenus (Élevé, Moyen, Faibles) • Actifs (Élevé, Moyen, Faibles) • Ratio dettes vs revenus (Élevé, Moyen, Faible) • Historique de paiement (Bon,mauvais) • Âge (moins de 25 ans, entre 25 et 50, entre 50 et 65, 65 et plus) Ultimement, Claude croit que la solvabilité de client(e)s dépend de: • leur fiabilité (Fiable, non fiable) • leur revenus futurs (Élevés, Moyens, Faibles) • leur ratio dettes vs revenus (Élevé, Moyen, Faible). L'expérience de Claude l'amène à croire que :

1. Un(e) client(e) avec un **bon historique de paiement** a tendance à être plus **fiable**;
2. Plus un(e) client(e) est **âgé(e)**, plus il/elle a de chance d'être **fiable**;
3. Les clients plus **âgés** ont tendance à avoir un fiable **ratio dettes vs revenus**;
4. La probabilité d'avoir un **bon historique de paiement** augmente au fur et à mesure que le **ratio de dette vs revenus diminue**;
5. Plus les **revenus** d'une personne sont élevés, plus cette personne a de chance d'avoir des **actifs** élevés;
6. Plus une personne a **d'actifs**, plus cette personne a de chance d'avoir un **revenu** élevé dans le futur;
7. Une personne **fiable** a tendance à être plus **solvable** qu'une personne non fiable;
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.

1 - Représentation graphique du réseau

```
dag_bank = dag(~Rev, ~Age, ~RDR|Age, ~HistP|RDR,
               ~Fiab|HistP:Age,
               ~Actif|Rev, ~RevF|Actif,
               ~Solv|RDR:Fiab:RevF)
plot(dag_bank)
```



2 - Élicitation des tables de probabilités conditionnelles

```
val0 = c("Faible","Moyen","Élevé") ## valeurs possibles pour chacune des variables
val1 = c("Mauvais","Bon")
val2 = c("Non Fiable","Fiable")
val3 = c("<25","[25:50]","[50:65]","[>65]")
val4 = c("Non Solvable","Solvable")

## ici code dépase la frontière de la page .pdf, vérifiez dans le fichier de rmd.
cp_Rev <- cptable(~Rev,values=c(33,33,33),levels=val0)
cp_Age <- cptable(~Age,values=c(25,25,25,25),levels=val3)
cp_RDR <- cptable(~RDR|Age,values=c(70,20,10,50,20,30,20,30,50,10,20,70),levels=val0)
cp_HistP <- cptable(~HistP|RDR, values = c(45,55,50,50,55,45),levels=val1)
cp_Actif <- cptable(~Actif|Rev,values = c(85,10,5,33,33,33,5,10,85),levels=val0)
cp_RevF <- cptable(~RevF|Actif,values = c(85,10,5,33,33,33,5,10,85),levels=val0)
cp_Fiab <- cptable(~Fiab|HistP+Age,values = c(80,20,60,40,40,60,20,80,80,20,60,40,40,60,20,80),levels=va
cp_Solv <- cptable(~Solv|RDR+Fiab+RevF,values = c(90,10,80,20,70,30,60,40,50,50,40,60,30,70,20,80,10,90


net_list = compileCPT(list(cp_Rev,cp_Age,cp_RDR,cp_HistP,cp_Actif,cp_RevF,cp_Fiab,cp_Solv))
grain_bank = grain(net_list)
```

1) P(Fiab = fiable | HistP = bon) > P(Fiab = non fiable | HistP = bon).

```r
print(querygrain(grain_bank, nodes=c("HistP","Fiab"), type="conditional"))
```

```
##          Fiab
## HistP     Non Fiable    Fiable
##   Mauvais  0.5977011 0.4047024
##   Bon      0.4022989 0.5952976
```

2) P(Age = Agé | Fiab = fiable) > P(Age = non Agé | Fiab = fiable).

```r
## ```supérieur à 65 représente agé donc P(Age = [>65] | Fiab = fiable) > P(Age = [50:65] | F = fiable)
## et P(Age = [>65] | Fiab = fiable) = P(Age = [25:50] | F = fiable) (cas particulier)
## , P(Age = [>65] | Fiab = fiable) > P(Age = [<25]| F = fiable)
## résumé : Plus un(e) client(e) est âgé(e), plus il/elle a de chance d’être fiable.
print(querygrain(grain_bank, nodes=c("Age","Fiab"), type="conditional"))
```

```
##          Fiab
## Age       Non Fiable    Fiable
##   <25      0.3468266 0.1530765
##   [25:50]  0.1489255 0.3511756
##   [50:65]  0.3513243 0.1485743
##   [>65]    0.1529235 0.3471736
```

3) P(Age = Agé | RDR = élevé) > P(Age = non Agé | RDR = élevé).

```r
## supérieur à  65 représente agé  donc P(Age = [>65] | RDR = élevé) > P(Age = [50:65] | RDR = élevé)
## , P(Age = [>65] | RDR = élevé) >  P(Age = [25:50] | RDR = élevé) et
## P(Age = [>65] | RDR = élevé) > P(Age = [<25] | RDR = élevé).
print(querygrain(grain_bank, nodes=c("Age","RDR"), type="conditional"))
```

```
##          RDR
## Age           Faible     Moyen  élevé
##   <25     0.46666667 0.2222222 0.0625
##   [25:50] 0.33333333 0.2222222 0.1875
##   [50:65] 0.13333333 0.3333333 0.3125
##   [>65]   0.06666667 0.2222222 0.4375
```

4) P(RDR = non élevé | HistP = bon) > P(RDR = élevé | Histp = bon).

```r
## P(RDR = Faible  | HistP = bon) > P(RDR = élevé | Histp = bon)
## et P(RDR = Faible  | HistP = bon) > P(RDR = Moyen | Histp = bon).
print(querygrain(grain_bank, nodes=c("RDR","HistP"), type="conditional"))
```

```
##         HistP
## RDR        Mauvais       Bon
##   Faible 0.3366584 0.4135338
##   Moyen  0.2244389 0.2255639
##   élevé  0.4389027 0.3609023
```

5) P(Rev = élevé | Actif = élevé) > P(Rev = non élevé | Actif = élevé).

```r
## P(Rev = élevé | Actif = élevé) > P(Rev = Faible | Actif = élevé)
## et P(Rev = élevé | Actif = élevé) > P(Rev = Moyen | Actif = élevé).
print(querygrain(grain_bank, nodes=c("Rev","Actif"), type="conditional"))
```

```
##         Actif
## Rev          Faible  Moyen      élevé
##   Faible 0.68918919 0.1875 0.04054054
##   Moyen  0.27027027 0.6250 0.27027027
##   élevé  0.04054054 0.1875 0.68918919
```

6) P(Actif = élevé | RevF = élevé) > P(Actif = non élevé | RevF = élevé).

```r
## P(Actif = élevé | RevF = élevé) > P(Actif = Moyen | RevF = élevé)
## et P(Actif = élevé | RevF = élevé) > P(Actif = Moyen | RevF = élevé).
print(querygrain(grain_bank, nodes=c("Actif","RevF"), type="conditional"))
```

```
##         RevF
## Actif        Faible     Moyen      élevé
##   Faible 0.81406385 0.2905759 0.04788611
##   Moyen  0.13805004 0.4188482 0.13805004
##   élevé  0.04788611 0.2905759 0.81406385
```

7) P(Fiab = fiable | Solv = Solvable) > P(Fiab = non fiable | Solv = Solvable).

```r
print(querygrain(grain_bank, nodes=c("Fiab","Solv"), type="conditional"))
```

```
##             Solv
## Fiab         Non Solvable  Solvable
##   Non Fiable    0.5928179 0.4086889
##   Fiable        0.4071821 0.5913111
```

8) P(RevF = élevé | Solv = Solvable) > P(RevF = fiable | Solv = Solvable).

```r
## P(RevF = élevé | Solv = Solvable) > P(RevF = fiable | Solv = Solvable)
## et P(RevF = élevé | Solv = Solvable) > P(RevF = Moyen | Solv = Solvable).
print(querygrain(grain_bank, nodes=c("RevF","Solv"), type="conditional"))
```

```
##         Solv
## RevF     Non Solvable Solvable
##   Faible    0.5590117 0.300918
##   Moyen     0.1409482 0.142009
##   élevé     0.3000401 0.557073
```

EXERCICE 2

```r
data(marks)

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


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 = (p2^x * (1-p2)^(1-x))^y * (p3^x * (1-p3)^(1-x))^(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)) )
  {
    a = ((p4^x * (1-p4)^(1-x))^y * (p5^x * (1-p5)^(1-x))^(1-y))^(z) *
      ((p6^x * (1-p6)^(1-x))^y * (p7^x * (1-p7)^(1-x))^(1-y))^(1-z)
  }
  return(a)
}

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

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

```r
L <- function(x1,x2,x3,x4,x5,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13){
  return(f1(x5,p1)*f2(x4,x5,p2,p3)*f3(x3,x4,x5,p4,p5,p6,p7)*
           f4(x2,x3,p8,p9)*f5(x1,x2,x3,p10,p11,p12,p13))
}
# = L(x1,x2,x3,x4,x5,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13)
```

```r
Nstat_1=0
NAnl_1_Stat_1=0
NAnl_1_Stat_0=0
NAnl_0_Stat_1=0
NAnl_0_Stat_0=0
NAlg_1_Anl_1_Stat_1=0
NAlg_1_Anl_0_Stat_1=0
NAlg_1_Anl_1_Stat_0=0
NAlg_1_Anl_0_Stat_0=0
NAlg_1=0
NVect_1_NAlg_1=0
NVect_1_NAlg_0=0
NVect_0_NAlg_1=0
NVect_0_NAlg_0=0
NMech_1_Vect_1_Alg_1=0
NMech_1_Vect_0_Alg_1=0
NMech_1_Vect_1_Alg_0=0
NMech_1_Vect_0_Alg_0=0
for(i in 1:88){
  if(notes_reussite[i,5]==1){
    Nstat_1=Nstat_1+1
  }
  if((notes_reussite[i,4]==1)&&(notes_reussite[i,5]==1)){
    NAnl_1_Stat_1=NAnl_1_Stat_1+1
  }
  if((notes_reussite[i,4]==1)&&(notes_reussite[i,5]==0)){
    NAnl_1_Stat_0=NAnl_1_Stat_0+1
  }
  if((notes_reussite[i,4]==0)&&(notes_reussite[i,5]==1)){
    NAnl_0_Stat_1=NAnl_0_Stat_1+1
  }
  if((notes_reussite[i,4]==0)&&(notes_reussite[i,5]==0)){
    NAnl_0_Stat_0=NAnl_0_Stat_0+1
  }
  if((notes_reussite[i,3]==1)&&(notes_reussite[i,4]==1)&&(notes_reussite[i,5]==1)){
    NAlg_1_Anl_1_Stat_1=NAlg_1_Anl_1_Stat_1+1
  }
  if((notes_reussite[i,3]==1)&&(notes_reussite[i,4]==0)&&(notes_reussite[i,5]==1)){
    NAlg_1_Anl_0_Stat_1=NAlg_1_Anl_0_Stat_1+1
  }
  if((notes_reussite[i,3]==1)&&(notes_reussite[i,4]==1)&&(notes_reussite[i,5]==0)){
    NAlg_1_Anl_1_Stat_0=NAlg_1_Anl_1_Stat_0+1
  }
```

```r
  if((notes_reussite[i,3]==1)&&(notes_reussite[i,4]==0)&&(notes_reussite[i,5]==0)){
    NAlg_1_Anl_0_Stat_0=NAlg_1_Anl_0_Stat_0+1
  }
  if(notes_reussite[i,3]==1){
    NAlg_1=NAlg_1+1
  }
  if((notes_reussite[i,2]==1)&&(notes_reussite[i,3]==1)){
    NVect_1_NAlg_1=NVect_1_NAlg_1+1
  }
  if((notes_reussite[i,2]==1)&&(notes_reussite[i,3]==0)){
    NVect_1_NAlg_0=NVect_1_NAlg_0+1
  }
  if((notes_reussite[i,2]==0)&&(notes_reussite[i,3]==1)){
    NVect_0_NAlg_1=NVect_0_NAlg_1+1
  }
  if((notes_reussite[i,2]==0)&&(notes_reussite[i,3]==0)){
    NVect_0_NAlg_0=NVect_0_NAlg_0+1
  }
  if((notes_reussite[i,1]==1)&&(notes_reussite[i,2]==1)&&(notes_reussite[i,3]==1)){
    NMech_1_Vect_1_Alg_1=NMech_1_Vect_1_Alg_1+1
  }
  if((notes_reussite[i,1]==1)&&(notes_reussite[i,2]==0)&&(notes_reussite[i,3]==1)){
    NMech_1_Vect_0_Alg_1=NMech_1_Vect_0_Alg_1+1
  }
  if((notes_reussite[i,1]==1)&&(notes_reussite[i,2]==1)&&(notes_reussite[i,3]==0)){
    NMech_1_Vect_1_Alg_0=NMech_1_Vect_1_Alg_0+1
  }
  if((notes_reussite[i,1]==1)&&(notes_reussite[i,2]==0)&&(notes_reussite[i,3]==0)){
    NMech_1_Vect_0_Alg_0=NMech_1_Vect_0_Alg_0+1
  }



}
p1=Nstat_1/88
p2=NAnl_1_Stat_1/Nstat_1
p3=NAnl_1_Stat_0/(88-Nstat_1)
p4=NAlg_1_Anl_1_Stat_1/NAnl_1_Stat_1
p5=NAlg_1_Anl_0_Stat_1/NAnl_0_Stat_1
p6=NAlg_1_Anl_1_Stat_0/NAnl_1_Stat_0
p7=NAlg_1_Anl_0_Stat_0/NAnl_0_Stat_0
p8=NVect_1_NAlg_1/NAlg_1
p9=NVect_1_NAlg_0/(88-NAlg_1)
p10=NMech_1_Vect_1_Alg_1/NVect_1_NAlg_1
p11=NMech_1_Vect_0_Alg_1/NVect_0_NAlg_1
p12=NMech_1_Vect_1_Alg_0/NVect_1_NAlg_0
p13=NMech_1_Vect_0_Alg_0/NVect_0_NAlg_0

#p1=P(Stat=1)
print(p1)
```

```
## [1] 0.3863636
```

```r
#p2=P(Anl=1|Stat=1)
print(p2)
```

```
## [1] 0.8529412
```

```r
#p3=P(Anl=1|Stat=0)
print(p3)
```

```
## [1] 0.5555556
```

```r
#p4=P(Alg=1|Anl=1,Stat=1)
print(p4)
```

```
## [1] 0.9655172
```

```r
#p5=P(Alg=1|Anl=0,Stat=1)
print(p5)
```

```
## [1] 0.8
```

```r
#p6=P(Alg=1|Anl=1,Stat=0)
print(p6)
```

```
## [1] 0.8333333
```

```r
#p7=P(Alg=1|Anl=0,Stat=0)
print(p7)
```

```
## [1] 0.4166667
```

```r
#p8=P(Vect=1|Alg=1)
print(p8)
```

```
## [1] 0.7761194
```

```r
#p9=P(Vect=1|Alg=0)
print(p9)
```

```
## [1] 0.2857143
```

```r
#p10=P(Mech=1|Vect=1,Alg=1)
print(p10)
```

```
## [1] 0.5576923
```

```r
#p11=P(Mech=1|Vect=0,Alg=1)
print(p11)
```

```
## [1] 0.3333333
```

```r
#p12=P(Mech=1|Vect=1,Alg=0)
print(p12)
```

```
## [1] 0.3333333
```

```r
#p13=P(Mech=1|Vect=0,Alg=0)
print(p13)
```

```
## [1] 0.1333333
```

```r
petoile <- function(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13) {
  P=1
  for(i in 1:88 ){
    Xi = notes_reussite[i,]
    P=P*L(Xi[1],Xi[2],Xi[3],Xi[4],Xi[5],p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13)
  }
  return (P)
}


petoile2 <- function(Plist) {
  P=1
  for(i in 1:88 ){
    Xi = notes_reussite[i,]
    #P=P*L(Xi[1],Xi[2],Xi[3],Xi[4],Xi[5],Plist[1],Plist[2],Plist[3],Plist[4],Plist[5],
          #Plist[6],Plist[7],Plist[8],Plist[9],Plist[10],Plist[11],Plist[12],Plist[13])
    P=P+log(L(Xi[1],Xi[2],Xi[3],Xi[4],Xi[5],Plist[1],Plist[2],Plist[3],Plist[4],Plist[5],
          Plist[6],Plist[7],Plist[8],Plist[9],Plist[10],Plist[11],Plist[12],Plist[13]))
  }
  return (-P) # on fait -P pour avoir argmax au lieux de argmin
}




notes_reussite[1,]
```

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

```r
#test = petoile(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.1,0.2,0.3,0.4,0.5,0.6)
#test

#x0 <- c(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13)
x0 <- c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)

solution <- optim(x0, petoile2)
print(solution$counts)
```

```
## function gradient
##      502       NA
```

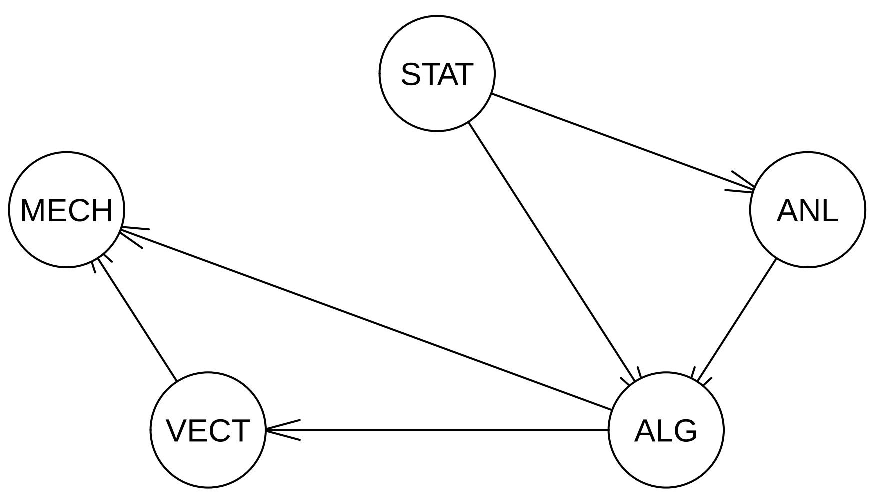
```
#la solution est déjà minimum
print(solution$par)
```

```
##  [1] 0.38589839 0.81888775 0.55986154 0.96479956 0.52862065 0.82099768
##  [7] 0.43109717 0.77892814 0.32375379 0.54137566 0.26393970 0.38264238
## [13] 0.08296025
```

```
print(solution$value)
```

```
## [1] 250.305
```

```
dag_notes = empty.graph(names(marks))
arcs(dag_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")))
plot(dag_notes)
```



```r
bn.fit(dag_notes, data = as.data.frame(notes_reussite))
```

```
##
##   Bayesian network parameters
##
##   Parameters of node MECH (Gaussian distribution)
##
## Conditional density: MECH | VECT + ALG
## Coefficients:
## (Intercept)         VECT          ALG
##   0.1282463    0.2178046    0.2101740
## Standard deviation of the residuals: 0.4756269
##
##   Parameters of node VECT (Gaussian distribution)
##
## Conditional density: VECT | ALG
## Coefficients:
## (Intercept)          ALG
##   0.2857143    0.4904051
## Standard deviation of the residuals: 0.4303528
##
##   Parameters of node ALG (Gaussian distribution)
##
## Conditional density: ALG | ANL + STAT
## Coefficients:
## (Intercept)          ANL         STAT
##   0.4504797    0.3558032    0.1872176
## Standard deviation of the residuals: 0.3752541
##
##   Parameters of node ANL (Gaussian distribution)
##
## Conditional density: ANL | STAT
## Coefficients:
## (Intercept)         STAT
##   0.5555556    0.2973856
## Standard deviation of the residuals: 0.4523587
##
##   Parameters of node STAT (Gaussian distribution)
##
## Conditional density: STAT
## Coefficients:
## (Intercept)
##   0.3863636
## Standard deviation of the residuals: 0.4897059
```

comparaison : on peut voir que dans les résultats :

STAT est 0.3863636 dans bn.fit et il est 0.3863636 dans paramétrisation, 0.38589839 dans vraisemblance il est très similaire donc correct

ANL | STAT est 0.5555556 dans bn.fit et il est 0.5555556 dans paramétrisation, 0.55986154 dans vraisemblance il est très similaire donc correct

ALG | ANL + STAT est 0.4504797 dans bn.fit et il est 0.4166667 dans paramétrisation, 0.43109717 dans vraisemblance il est très similaire donc correct

VECT | ALG est 0.2857143 dans bn.fit et il est 0.2857143 dans paramétrisation, 0.26393970 dans vraisemblance il est très similaire donc correct

MECH | VECT + ALG est 0.1282463 dans bn.fit et il est 0.1333333 dans paramétrisation, 0.08296025 dans vraisemblance il est un peu similaire donc correct

EXERCICE 3

Question 1

Vous trouverez la rédaction à cette question dans le whiteboard en pièce jointe.

Nous avons utilisé 7 lois normales de probabilités conditionelles respectant la structure du réseau bayésien proposé. Le produit de celles-ci représente la probabilité jointe de toutes les variables gaussiennes

Il y a 24 paramètres à ajutster : 17 "alphas" et 7 "sigmas".

```
load.Rdata( filename="ApportCalorique.RData", "dat.s3" )
xtest = head(dat.s3)
summary(dat.s3)
```

```
##       Phys            IMC              C                B       
##  Min.   :17.93   Min.   :18.76   Min.   :10.98   Min.   : 636  
##  1st Qu.:44.47   1st Qu.:23.56   1st Qu.:21.52   1st Qu.:1008  
##  Median :50.76   Median :24.91   Median :24.15   Median :1096  
##  Mean   :50.33   Mean   :24.94   Mean   :24.20   Mean   :1097  
##  3rd Qu.:56.02   3rd Qu.:26.34   3rd Qu.:26.81   3rd Qu.:1191  
##  Max.   :79.29   Max.   :31.17   Max.   :36.33   Max.   :1500  
##        H                Ap              A       
##  Min.   :0.4400   Min.   :3.950   Min.   :1671  
##  1st Qu.:0.8700   1st Qu.:5.888   1st Qu.:2331  
##  Median :0.9900   Median :6.430   Median :2486  
##  Mean   :0.9973   Mean   :6.451   Mean   :2488  
##  3rd Qu.:1.1225   3rd Qu.:7.003   3rd Qu.:2636  
##  Max.   :1.5000   Max.   :9.100   Max.   :3240  
```

```
var(dat.s3)
```

```
##              Phys          IMC             C            B            H
## Phys   75.3969131  -0.51513356 -4.095558e+00   15.9460781  0.714101402
## IMC    -0.5151336   4.01599005  1.145146e+00  133.7411433 -0.016479014
## C      -4.0955576   1.14514617  1.675010e+01   58.8141868 -0.004031665
## B      15.9460781 133.74114332  5.881419e+01 17963.5995977 -0.381786854
## H       0.7141014  -0.01647901 -4.031665e-03   -0.3817869  0.033332580
## Ap      4.3771775   0.10714156 -1.600525e-02   23.1313388  0.146669289
## A    1857.7198710 -16.85802508 -3.276121e+02 1664.5275137 27.618098213
##                Ap           A
## Phys   4.37717751  1857.71987
## IMC    0.10714156   -16.85803
## C     -0.01600525  -327.61207
## B     23.13133878  1664.52751
## H      0.14666929    27.61810
## Ap     0.71317182   150.42096
## A    150.42095605 53478.50180
```

```r
gaussienneUnivariée <- function(x,mu,sigma){

  xdoubele = as.numeric(unlist(x))
  mudouble = as.numeric(unlist(mu))
  sigmadouble = as.numeric(unlist(sigma))

  return(dnorm(xdoubele, mudouble, sigmadouble))

}


#Bien repasser sur quelles données on passe en entrée de dnorm

N1 <- function(x,alphaPHY,sigmaPHY){
  return(gaussienneUnivariée(x, alphaPHY, sigmaPHY))
}

N2 <- function(x,alphaIMC,sigmaIMC){
  return(gaussienneUnivariée(x,alphaIMC, sigmaIMC))
}

N3 <- function(x,x_IMC,x_Phys,alphaC,alphaCI,alphaCP,sigmaCIP){
  return(gaussienneUnivariée(x, alphaC+alphaCI*x_IMC+alphaCP*x_Phys, sigmaCIP))
}

N4 <- function(x,x_IMC,alphaB,alphaBI,sigmaBI){
  return(gaussienneUnivariée(x, alphaB+alphaBI*x_IMC, sigmaBI))
}


N5 <- function(x,x_Phys,alphaH,alphaHP,sigmaHP){
  return(gaussienneUnivariée(x, alphaH+alphaHP*x_Phys, sigmaHP))
}


N6 <- function(x,x_B,x_H,x_Phys,alphaAP,alphaAPB,alphaAPH,alphaAPP,sigmaAPBHP){
  return(gaussienneUnivariée(x, alphaAP+alphaAPB*x_B+alphaAPH*x_H+alphaAPP*x_Phys, sigmaAPBHP))
}

N7 <- function(x,x_C,x_AP,x_Phys,alphaA,alphaAC,alphaAAP,alphaAPh,sigmaACAPP){
  return(gaussienneUnivariée(x, alphaA+alphaAC*x_C+alphaAAP*x_AP+alphaAPh*x_Phys, sigmaACAPP))
}


vraisemblancheEx3 <- function(
  x1,x2,x3,x4,x5,x6,x7,alphaPHY,sigmaPHY,alphaIMC,sigmaIMC,
          alphaC,alphaCI,alphaCP,sigmaCIP,alphaB,alphaBI,sigmaBI,alphaH,alphaHP,
          sigmaHP,alphaAP,alphaAPB,alphaAPH,alphaAPP,sigmaAPBHP,
          alphaA,alphaAC,alphaAAP,alphaAPh,sigmaACAPP){

  a = N1(x2,alphaPHY,sigmaPHY)*N2(x1,alphaIMC,sigmaIMC)*
    N3(x4,x1,x2,alphaC,alphaCI,alphaCP,sigmaCIP)*
    N4(x3,x1,alphaB,alphaBI,sigmaBI)*N5(x5,x2,alphaH,alphaHP,sigmaHP)*
```

```r
    N6(x6,x3,x4,x2,alphaAP,alphaAPB,alphaAPH,alphaAPP,sigmaAPBHP)*
    N7(x7,x4,x6,x2,alphaA,alphaAC,alphaAAP,alphaAPh,sigmaACAPP)
  return(a)
}

petoileExo3 <- function(AlphaSiglist)
{
  P=0
  for(i in 1:1000 ){
    Xi = dat.s3[i,]
    P=P+log(1+vraisemblancheEx3(
      Xi[1],Xi[2],Xi[3],Xi[4],Xi[5],Xi[6],Xi[7],
      AlphaSiglist[1],AlphaSiglist[2],AlphaSiglist[3],
      AlphaSiglist[4],AlphaSiglist[5],AlphaSiglist[6],
      AlphaSiglist[7],AlphaSiglist[8],AlphaSiglist[9],
      AlphaSiglist[10],AlphaSiglist[11],
      AlphaSiglist[12],AlphaSiglist[13],AlphaSiglist[14],
      AlphaSiglist[15],AlphaSiglist[16],
      AlphaSiglist[17],AlphaSiglist[18],AlphaSiglist[19],
      AlphaSiglist[20],AlphaSiglist[21],
      AlphaSiglist[22],AlphaSiglist[23],AlphaSiglist[24]))
  }
  # on fait log(1+x) pour que le fichier compile même pour
  #des valeurs négatives de x. Cela marche comme la log
  #vraissemblance car log(1+x) est strictement croissant.
  return (-P) # on fait -P pour avoir argmax au lieux de argmin
}

x0 <- c(50.33,sqrt(75),24.94,sqrt(4),24.2,24.2,24.2,sqrt(1),1097,561,sqrt(134),
        1,1,sqrt(0.7),2488,2488,2488,2488,sqrt(23),2488,2488,2488,2488,sqrt(0))
# ici on a pris les moyennes et les équarts
# type trouvés en début de question dans "summary" et "var".


solution <- optim(x0, petoileExo3)
print(solution$counts)
```

```
## function gradient
##       25       NA
```

```r
print(solution$par)
```

```
##  [1]   50.330000    8.660254   24.940000    2.000000   24.200000   24.200000
##  [7]   24.200000    1.000000 1097.000000  561.000000   11.575837    1.000000
## [13]    1.000000    0.836660 2488.000000 2488.000000 2488.000000 2488.000000
## [19]    4.795832 2488.000000 2488.000000 2488.000000 2488.000000    0.000000
```

```r
print(solution$value)
```

```
## [1] 0
```

Interprétation : On peut remarquer que les résultats des optimisations n'ont pas changés par rapport aux **x0** entrées, Ceci montre qu'utiliser les moyennes et les écarts qu'on a trouvés dans « **summary(dat.s3)** » et « **var(dat.s3)** » soit une bonne idée. Cela prouve également qu'il s'agit d'un bon modèle pour maximiser la vraisemblance. Considérons un échantillon théorique **(X__1,. . . ,X__7)** de la loi **f__x(x)**. Estimons un paramètre par la méthode du maximum de vraisemblance, afin de trouver la paramétrisation qui permet de rendre la vraisemblance maximale. Dans le cas si on observe les données comme réalisation d'un échantillon de la loi ** f_x(x)**. Résumé des interactions : B de dépend IMC, C de dépend IMC,Phys, H de dépend Phys, Ap de dépend B,H,Phys, A de dépend C,Ap,Phys, IMC et Phy dépendent de rien. c'est peut être normal de trouver des résultats similaires. Parce qu'on sait que pour mu qui maximise la vraisemblance, c'est les moyennes. Et que la matrice de covariance qui maximise, c'est la distance Mahalanobis.

$$\mathbb{P}(IMC, phys, B, C, H, A_p, A) = \mathbb{P}(IMC) \cdot \mathbb{P}(phys) \cdot \mathbb{P}(B \mid IMC) \cdot \mathbb{P}(C \mid IMC, phys)$$
$$\cdot \mathbb{P}(H \mid phys) \cdot \mathbb{P}(A_p \mid B, H, phys) \cdot \mathbb{P}(A \mid C, A_p, phys)$$

$$L = N(\alpha_{IMC}, \sigma^2_{IMC}) \cdot N(\alpha_{phys}, \sigma^2_{phys}) \cdot N(\alpha_B + \alpha_{BIMC} IMC, \sigma^2_{IMC})$$
$$\cdot N(\alpha_C + \alpha_{CIMC} IMC + \alpha_{Cphys} phys, \sigma^2_{CIP}) N(\alpha_H + \alpha_{Hp} phys, \sigma^2_{Hp})$$
$$\cdot N(\alpha_{Ap} + \alpha_{ApB} \cdot B + \alpha_{ApH} H + \alpha_{Apphy} \cdot phys, \sigma^2_{ABHp})$$
$$\cdot N(\alpha_A + \alpha_{AC} C + \alpha_{AA_p} A_p + \alpha_{Ap} phys, \sigma^2_{ACA_pp})$$

$$V = \prod_{i=1}^{n} L(\ldots)$$