STT760 - Devoir 1 - Partie pratique

Remise: 25 octobre 2023

Instructions relatives à la remise du devoir.

La rédaction du devoir peut se faire en équipe d'au plus 5 personnes. Vos solutions devront être rédigées par ordinateur et placées dans un seul et même fichier incluant le nom de chaque membre de l'équipe, et devront être envoyées par courriel à l'adresse courriel de la personne chargée de la correction pour ce cours (adresse qui vous sera communiquée sous peu) avant 17h le 25 octobre 2023. Indiquez clairement le numéro de l'exercice auquel vos solutions réfèrent.

Mise en contexte

L'objectif de ce travail est de mettre en pratique des notions liées aux réseaux bayésiens tout en s'initiant au logiciel **R**. En plus de l'introduction complète au logiciel **R** par Venables and Smith (2004) disponible à l'adresse http://cran.r-project.org/doc/manuals/R-intro.pdf, des documents complémentaires ont été placé sur la plateforme **Teams** du cours, dont un excellent tutoriel monté par Pascal Neveu de l'Inra supagro qui m'a enseigné il y a quelques années, et une introduction que j'ai montée avec ma superviseure de post-doctorat Aurore Delaigle à l'université de Melbourne dans le cadre de notre enseignement conjoint du cours *Statistics for Bioinformatics*.

L'interface **RStudio** est fortement recommandée, dont le téléchargement est disponible gratuitement via le site https://rstudio.com/products/rstudio/download/.

Réchauffement - représentation d'un réseau bayésien

Le but de cet exercice est de créer un réseau bayésien sous **R** pour représenter le système entourant l'exemple du gicleur détaillé dans le chapitre 3 des notes de cours. Cet exemple fait intervenir cinq variables:

- G: l'état du gazon (0=sec, 1=mouillé)
- P: l'événement "il a plu" (0=non, 1=oui)
- O: l'état du gicleur (0=fermé, 1=allumé)
- C: l'état du ciel (0=ensoleillé, 1=nuageux)

Installer et charger les librairies nécessaires

Sous **RStudio**, créez un nouveau script en utilisant en choisissant `File -> New File -> R Script`.

Exécutez les commandes ci-dessous.

```r
if (!requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager")


BiocManager::install(c("gRain", "gRbase","graph","RBGL","Rgraphviz"))

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

La librairie **graph** permet la représentation d'un graph dans **R** comme un object de type *GraphNEL* constitué d'une liste de noeuds et d'arêtes (Graph as Nodes and Edges List). La librairie **GRbase** s'inscrit en complément de **graph** et permet d'effectuer des opérations sur les objets de tpe **GraphNEL**. La librairie **Rgraphviz** permet de produire des représentations graphiques des objets **GraphNEL**. La librairie **RBGL** inclut quelques algorithmes sur les objects **GraphNEL**.

Création d'un objet GraphNel

Un graphe peut être créer suivant la commande *dag()* de la façon suivante.

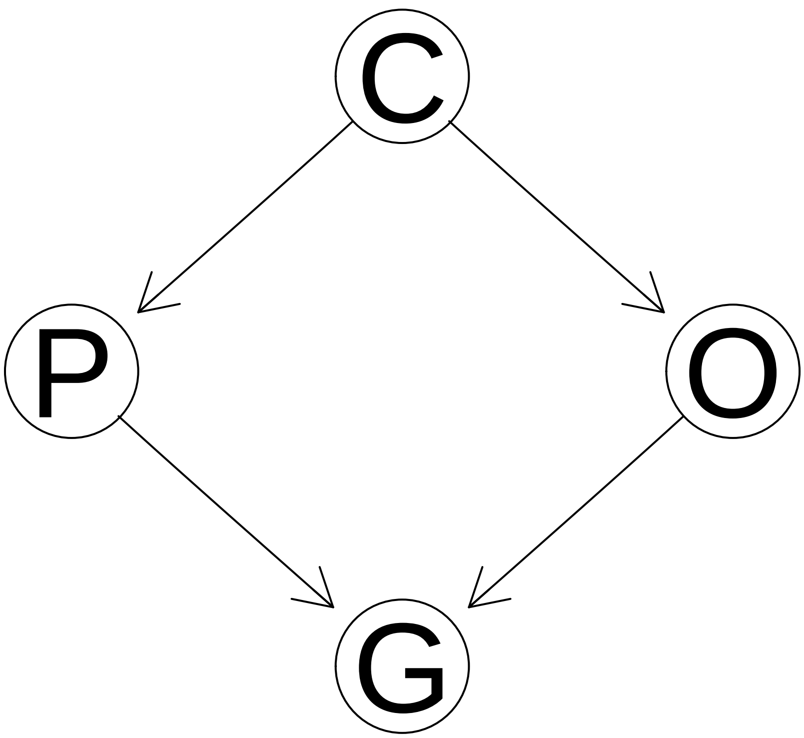
```
dag_gicleur = dag(~C, ~P|C, ~O|C, ~G|O:P)
dag_gicleur
```

```
## A graphNEL graph with directed edges
## Number of Nodes = 4
## Number of Edges = 4
```

Dans l'énoncé ci-dessus, $\sim C$ signifie que C n'a pas de noeud parent, $P|C$ signifie que P a C pour parent, alors que $\sim G|O : P$ signifie que G a O et P comme parents. Il est possible de créer le même objet **GraphNEL** a l'aide d'énoncés équivalents, voir l'aide associée à la fonction **dag()** en exécutant la commande **?dag**.

Le graphe orienté résultant peut alors être visualisé via l'exécution suivante.

```
plot(dag_gicleur)
```



Il est possible de représenter un graphe orienté à l'aide d'une matrice d'adjacence $[M]$ dont les entrées sont des 0 ou des 1.

```
Adj_gicleur = as(dag_gicleur,"matrix")
Adj_gicleur
```

```
##   C P O G
## C 0 1 1 0
## P 0 0 0 1
```

```
## O 0 0 0 1
## G 0 0 0 0
```

La d-séparation peut-être vérifiée à l'aide de la fonction *dSep()* de la librairie **ggm**.

```r
#install.packages("ggm")
library("ggm")
dSep(Adj_gicleur,"P","O","G")
```

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

```r
dSep(Adj_gicleur,"P","O","C")
```

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

Création et interrogation d'un réseau bayésien

Un réseau bayésien est la donnée d'un graphe orienté et d'une distribution de probabilité se factorisant suivant le graphe. Cette dernière peut être spécifiée à l'aide de l'élicitation de la loi conditionnelle de chacun des noeuds étant donné ses parents. Dans l'exemple du gicleur, nous avons vu qu'il fallait ainsi spécifier:

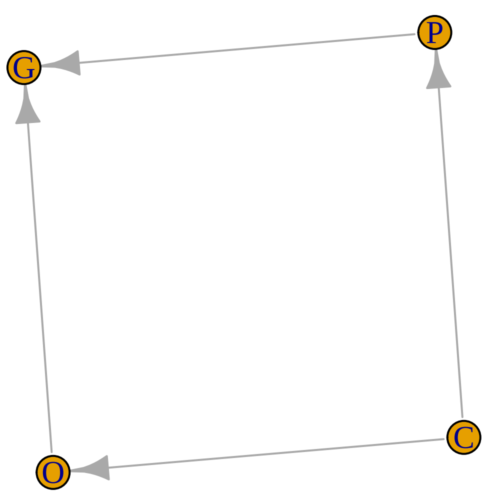
- $\mathbb{P}(G \mid O, P)$
- $\mathbb{P}(P \mid C)$
- $\mathbb{P}(O \mid C)$
- $\mathbb{P}(C)$.

Dans ce qui suit, nous verrons comment créer un réseau bayésien en spécifiant ces probabilités avec la fonction *cptable()* de la librairie **gRain**.

```r
library("gRain")
val = c("0","1") ## valeurs possibles pour chacune des variables
cp_C <- cptable(~C,values=c(50,50),levels=val)
cp_P <- cptable(~P|C,values=c(90,10,5,95),levels=val)
cp_O <- cptable(~O|C, values = c(40,60,55,45),levels=val)
cp_G <- cptable(~G|P+O,values = c(1,0,0,1,0,1,0,1),levels=val)
```

Le réseau peut alors être créé avec la fonction *compileCPT* de la librairie **gRain**. On crée ensuite un objet de type ***grain*** (GRAphical Independence Network).

```r
library("gRain")
net_list = compileCPT(list(cp_C,cp_P,cp_O,cp_G))
grain_gicleur = grain(net_list)
plot(grain_gicleur$dag)
```



Un objet de type *grain* peut alors être utilisé pour le calcul de probabilité. Par exemple, les commandes ci-dessous permettent de calculer respectivement la probabilité marginale que le gazon soit mouillé et la probabilité qu'il ait plu sachant l'état du gazon.

```
querygrain(grain_gicleur, nodes="G", type="marginal")
```

```
## $G
## G
##       0       1
## 0.19375 0.80625
```

```
querygrain(grain_gicleur, nodes=c("P","G"), type="conditional")
```

```
##    G
## P   0         1
##   0 1 0.3488372
##   1 0 0.6511628
```

Il est aussi possible d'interroger la distribution du graphe étant donné un événement fixe. Pour ce faire, il faut d'abord mettre à jour le graphe et la distribution de probabilité étant donné l'événement.

```
grain_gicleur_mouille = setFinding(grain_gicleur, nodes=c("G","C"),states=c("1","1"))
querygrain(grain_gicleur_mouille, nodes=c("P","O"), type="joint")
```

```
##    O
## P           0          1
##   0 0.0000000 0.02313625
##   1 0.5372751 0.43958869
```

Une structure alternative

La représentation d'un réseau bayésien par l'entremise d'un objet *GraphNEL* est assez flexible pour permettre une variété de modèles, et fonctionne de concert avec les algorithmes implémentés dans les librairies **gRain** et **gRbase**. Ces librairies ciblent l'apprentissage de paramètres et le calcul de probabilités à partir d'un réseau spécifié, mais ne permettent pas l'apprentissage de la structure du réseau, ce que la librairie **bnlearn** (Bayesian Network LEARNing) permet.

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

Dans ce qui suit, considérons un exemple originellement traité dans Mardia et al. (1979), Whittaker (1990) et Edwards (2000), puis repris dans plusieurs ouvrages de références au sujet de réseaux bayésiens. Cet exemple sous-tend les résultats de 88 personnes étudiantes à des examens réalisés dans cinq sujets différents: en mécanique (MECH), en algèbre (ALG), en algèbre vectorielle (VECT), en analyse (ANL) et en statistique (STAT). Dans chacune de ces matières, les résultats se trouvent entre 0 et 100. Les données sont incluses dans la librairie **bnlearn**.

```
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
```

La librairie **bnlearn** représente un réseau bayésien par l'entremise d'un objet de type *bn*. Un objet *bn* est créé en construisant d'abord un graphe sans arête à partir d'une liste de noeuds, puis en le modifiant subséquamment pour y ajouter des arêtes.

```r
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")))
dag_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
```

Comme c'était le cas pour les objets de type *GraphNEL*, on peut spécifier les paires d'arêtes en élicitant la matrice d'adjacence associée au graphe.

```r
mat_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)))
mat_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
```

La strucutre du réseau est alors créée de façon similaire.

```
dag2_notes = empty.graph(names(marks))
amat(dag2_notes) = mat_ad
all.equal(dag_notes, dag2_notes)
```

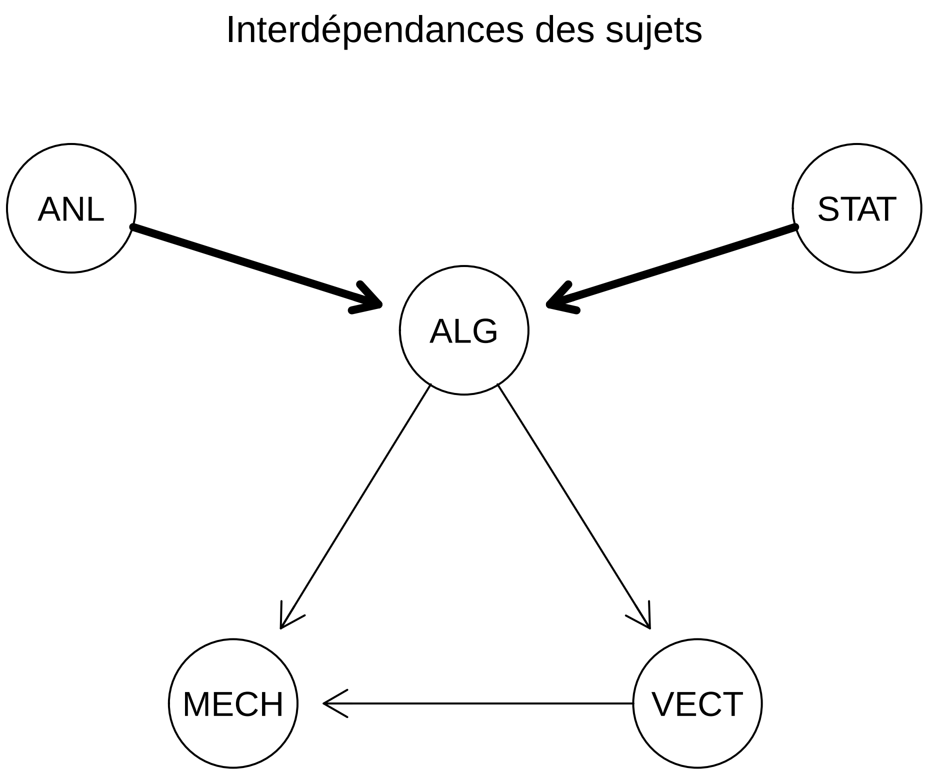
```
## [1] TRUE
```

Il est possible de modifier un objet *bn* pour y ajouter (**set.arc**) ou y enlever (**drop.arc**) des arêtes.

```
dag2 = drop.arc(dag_notes, from = "STAT", to = "ANL")
```

La structure se visualise avec l'aide de la commande suivante.

```
v_st = list(arcs = vstructs(dag2, arcs = TRUE), lwd = 4, col = "black")
graphviz.plot(dag2, highlight = v_st, layout = "fdp", main = "Interdépendances des sujets")
```



Exercice 1 (Inspiré de Koller, Coursera 2019) - 10 points

Mise en situation

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.

Objectif

Construire un réseau bayésien cohérent avec la mise en situation ci-dessus.

Instructions

Vous devrez :

1. Produire une représentation graphique de votre réseau.
2. Produire une élicitation des tables de probabilités conditionnelles associées à votre réseau qui respecte les points 1 à 8 ci-dessus. Par exemple, si F et H sont des variables aléatoires représentant respectivement la fiabilité la qualité de l'historique de paiement, le point 1 indique que

$$\mathbb{P}(F = \text{fiable} \mid H = \text{bon}) > \mathbb{P}(F = \text{fiable} \mid H = \text{mauvais}).$$

Pour ce faire, vous devrez créer un objet de type **graphNEL**. Ensuite, vous devrez produire des commandes qui permmettent de vérifier chacun des points 1 à 8 ci-dessus.

Pour cet exercice, vous devrez remettre la représentation graphique de votre réseau ainsi que le code qui vous a permis de répondre aux instructions 1 et 2.

Exercice 2 - 10 points

Mise en contexte

Considérons le jeu de données **marks** introduit précédemment qui contient des résultats d'évaluations d'élèves dans cinq matières différentes. Alexis, qui s'occupe de la direction d'un programme de mathématique, souhaite ajuster un réseau bayésien pour quantifier les interdépendances de manière probabiliste dans la réussite aux évaluations des élèves de son programme. Il croit que la structure de réseau bayésien ***dag_notes*** représente fidèlement ces interdépendances.

Objectif

Ajuster les paramètres du réseau bayésien correspondant, sachant que le seuil de réussite pour chacun des cours est fixé à 45%.

Indications

1. Préparer les données en exécutant la commande

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

Ainsi, la matrice **notes_reussite** contient des E et des R, où un R indique une réussite et un E indique un échec.

2. Soit $\mathbf{X}_i = (X_{i1}, \ldots, X_{i5})$ un vecteur aléatoire tel que X_{i1} (respectivement $X_{i2} \ldots, X_{i5}$) vaut 1 si l'étudiant(e) i a réussi mécanique (resp. algèbre vectorielle, algèbre, analyse et statistique). Alors,

$$\mathbb{P}(\mathbf{X}_i) = \mathbb{P}(X_{i5})\mathbb{P}(X_{i4} \mid X_{i5})\mathbb{P}(X_{i3} \mid X_{i4}, X_{i5})\mathbb{P}(X_{i2} \mid X_{i3})\mathbb{P}(X_{i1} \mid X_{i2}, X_{i3})\,.$$

   Le calcul de $\mathbb{P}(\mathbf{X}_i)$ requiert donc de spécifier cinq fonctions, telles que

$$f_1(x, p_1) = p_1^x \times (1 - p_1)^{1-x},$$

   où $p_1 = \mathbb{P}(X_{i5} = 1)$ et $x \in \{0, 1\}$ et

$$f_2(x, y, p_2, p_3) = (p_2^x \times (1 - p_2)^{1-x})^y \times (p_3^x \times (1 - p_3)^{1-x})^{1-y},$$

   où $p_2 = \mathbb{P}(X_{i4} = 1 \mid X_{i5} = 1)$, $p_3 = \mathbb{P}(X_{i4} = 1 \mid X_{i5} = 0)$ et $x, y \in \{0, 1\}$.

Ce type de fonctions peut être créé sous **R** en suivant la structure ci-dessous:

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

De cette manière, créez des fonctions *f1*,...,*f5* qui permettront de paramétriser $\mathbb{P}(\mathbf{X}_i)$. Puis, construisez une fonction

$$L(x_1, \ldots, x_5, p1, \ldots, p13)$$

qui calcule $\mathbb{P}(\mathbf{X}_i = (x_1, \ldots, x_5))$ en fonction de la spécification du vecteur de paramètres $\mathbf{p} = (p1, \ldots, p13)$.

3. Il est possible d'ajuster les paramètres du modèle en utilisant la méthode du maximum de vraisemblance. Formellement, étant donné un échantillon d'observations indépendantes et de même distribution, cette méthode prescrit de choisir $\mathbf{p}^\star$ tel que

$$\mathbf{p}^\star = \text{ arg min } \prod_{i=1}^{n} L(\mathbf{X}_i, \mathbf{p}).$$

   Vous devrez donc produire un script qui permet de calculer la valeur de $\mathbf{p}^\star$ à partir de la matrice **notes_reussite**. Pour ce faire, vous pouvez d'abord calculer analytiquement la valeur de $\mathbf{p}^\star$, puis implémenter la solution.

4. Comparez votre solution avec le résultat de la commade suivante:

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

Vous devrez me remettre votre scripte ainsi que les éléments d'analyse vous ayant permis de répondre aux instructions 1 à 4.