

Non parametric inference of dependence structures in high dimension with graphical models

an aGrUM/OpenTURNS interaction (WIP)

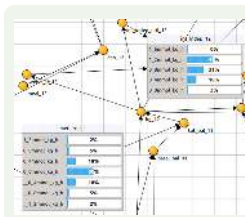
Pierre-Henri Wuillemin (LIP6)/Régis Lebrun (AIRBUS) – 2018

BN&Copules

- 1 Modèle graphique
 - Réseau bayésien
 - apprentissage
- 2 aGrUM
 - Modèle graphiques
 - motivation
 - librairie aGrUM
 - Applications
- 3 module openTURNS-aGrUM
- 4 Modèles graphiques et copules
 - Copules in a nutshell
 - Décomposition de Vines
 - Cumulative Distribution Network
 - Copula Bayesian Networks
- 5 Apprentissage de modèle graphique dans les copules
 - Implémentation

OpenTURNS'18 

2 / 57



Modèles graphiques
discrets

Modèle probabiliste complexe discret

La représentation probabiliste d'un système est caractérisé par un univers Ω où chaque $\omega \in \Omega$ est un état du système.

Un **système complexe** est caractérisé par un univers Ω de grande taille.

➡ Définition (Modèle factorisé)

- Un modèle probabiliste (sur Ω) est **factorisé** lorsqu'il existe une famille $\mathcal{X} = (X_i)_{i \leq n}$ de variables aléatoires sur Ω telle que chaque $\omega \in \Omega$ est caractérisé de manière unique par les valeurs $(X_i(\omega))_{i \leq n}$.
- Dans un modèle factorisé, une probabilité sur Ω sera donc représentée par une loi **jointe** des variables de \mathcal{X} .

$$\forall \omega \in \Omega, p(\omega) = p(X_1 = X_1(\omega), X_2 = X_2(\omega), \dots, X_n = X_n(\omega))$$



Explosion combinatoire : Si toutes les variables sont binaires, un système factorisé en n variables nécessitent $\approx 2^n$ valeurs !

La factorisation peut-elle permettre d'améliorer la compacité ? Grâce à l'**indépendance conditionnelle** !!

$$2^3 \quad p(X, Y, Z) = p(X) \cdot p(Y | X) \cdot p(Z | X, Y) \quad 2 + 2^2 + 2^3$$

Avec $X \perp\!\!\!\perp Y$ et $Z \perp\!\!\!\perp X, Y$:

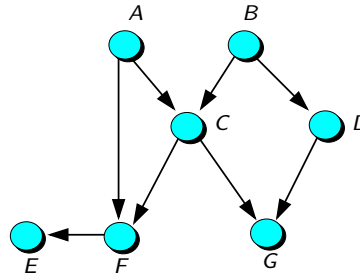
$$2^3 \quad p(X, Y, Z) = p(X) \cdot p(Y) \cdot p(Z) \quad 2 + 2 + 2$$

OpenTURNS'18

4 / 57

Indépendances ... et graphe

$$\begin{aligned} p(A, B, C, D, E, F, G) = & p(A) \cdot P(B|A) \cdot P(C|A, B) \\ & \cdot P(D|A, B, C) \cdot P(E|A, B, C, D) \\ & \cdot P(F|A, B, C, D, E) \cdot P(G|A, B, C, D, E, F) \end{aligned}$$



Graphoïde, I-map, Local Markov Property

- "U indépendant de V conditionnellement à W" et "U est séparé de V par W" partagent une même structure : **graphoïde**
- \Rightarrow utilisation d'un graphe pour représenter les indépendances dans $p()$:

Independency-MAP

- **Local Markov Property** : $X \perp\!\!\!\perp \text{non-descendant}(X) | \text{parents}(X)$

Par exemple,

$$G \perp\!\!\!\perp E, F, A, B | C, D$$

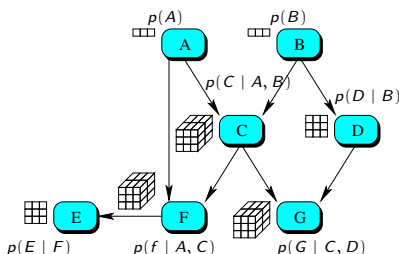
OpenTURNS'18

Modèle graphique

Réseau bayésien

5 / 57

réseau bayésien : exemple et définition



$$\begin{aligned} p(A, B, C, D, E, F, G) = & p(A) \cdot p(B) \\ & \cdot p(C | A, B) \cdot p(D | B) \cdot p(F | A, C) \\ & \cdot p(E | F) \cdot p(G | C, D) \end{aligned}$$

Tout se passe comme si l'information était localisée dans les nœuds !

$$P(A, B, C, D, E, F, G) : \quad 3^7 = 2187 \text{ paramètres} \quad \text{vs} \quad 105 \text{ paramètres dans le BN !}$$

➡ Définition (Réseau bayésien (BN))

Un réseau bayésien est une représentation compacte d'une distribution de probabilité sur un ensemble de variables aléatoires. Il s'appuie sur un graphe orienté sans circuit (DAG) pour représenter son modèle d'indépendance. La décomposition de la loi jointe suivant le graphe s'écrit :

$$P(\mathcal{X}) = \prod_i P(X_i | \Pi_i)$$

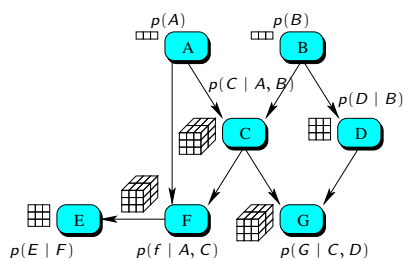
Modèle graphique

Réseau bayésien

6 / 57

Utilisations des BNs : inférences probabilistes

diagnostic : $P(A | F)$



- diagnostic de panne
- sûreté de fonctionnement
- filtrage de spams

prédiction $P(E | B, A)$

- Simulation de process (industriels)
- prévisions boursières
- modélisation de joueurs

Autres tâches

- Cas le plus probable (MPE) : $\arg \max P(\mathcal{X} | D)$
- Analyse de sensibilité, information mutuelle, etc.
- Troubleshooting : $\arg \max \frac{P(\cdot)}{C(\cdot)}$

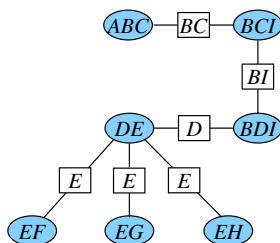
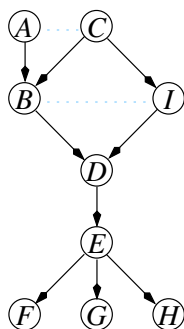
OpenTURNS'18

Modèle graphique

Réseau bayésien

7 / 57

Inférence : Arbre de jonction, arbre de cliques



➡ Définition (Décomposition suivant l'arbre de cliques)

La décomposition de la loi jointe suivant le graphe s'écrit :

$$P(\mathcal{X}) = \frac{\prod_{C \in \text{clique}(G)} P(C)}{\prod_{S \in \text{separateur}(G)} P(S)}$$

OpenTURNS'18

Modèle graphique

Réseau bayésien

8 / 57

Apprentissage de modèles graphiques - cadre général

Tableau général de l'apprentissage

- Recherche de relation symétrique + orientation (causalité)
 - algorithme IC/PC
 - algorithme IC*/FCI
- Recherche heuristique (score)
 - Dans l'espace des structures (BN ou équivalent de Markov),
 - algorithmes essayant de maximiser un score (entropie, AIC, BIC, MDL, BD, BDe, BDeu, ...).

OpenTURNS'18

Modèle graphique

apprentissage

9 / 57

Recherche de relation symétrique

En terme statistique, les relations testables sont symétriques : **corrélation ou indépendance entre variables aléatoires**.

Par contre, une fois des relations 2 à 2 trouvées, il s'agit de tester certaines indépendances conditionnelles (V-structure) qui forcent les orientations.

Principe de base (IC, IC*, PC, FCI)

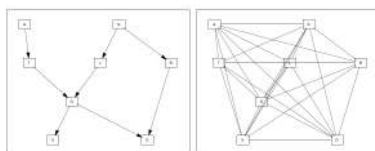
- 1 Construire le graphe (non orienté) des relations de dépendance trouvées statistiquement (χ^2 ou autre) :
 - Ajouter des arêtes à partir du graphe vide.
 - Retirer des arêtes à partir du graphe complet.
- 2 Détecter les V-structures et les orientations qu'elles impliquent.
- 3 Finaliser les orientations en restant dans la même classe d'équivalence de Markov.

Écueils principaux : un très grand nombre de tests d'indépendances, chaque test étant très sensible au nombre de données disponibles.

Exemple PC

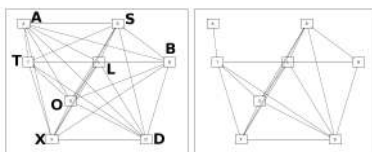
- Soit un réseau bayésien (à gauche) qui a permis de créer une base de 5000 cas (Exemple de Philippe Leray).

Étape 0 : Graphe non orienté reliant tous les nœuds.



- Par des χ^2 , on teste toutes les indépendances marginales ($X \perp\!\!\!\perp Y$) puis les indépendances par rapport à une variable ($X \perp\!\!\!\perp Y | Z$).

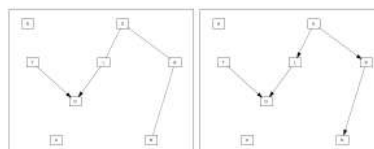
Étape 1a : Suppression des ind. conditionnelles d'ordre 0



On trouve : $A \perp\!\!\!\perp S$, $L \perp\!\!\!\perp A$, $B \perp\!\!\!\perp A$, $O \perp\!\!\!\perp A$, $X \perp\!\!\!\perp A$, $D \perp\!\!\!\perp A$,

$T \perp\!\!\!\perp S$, $L \perp\!\!\!\perp T$, $O \perp\!\!\!\perp B$, $X \perp\!\!\!\perp B$.

Étape 4 : Instanciation du PDAG



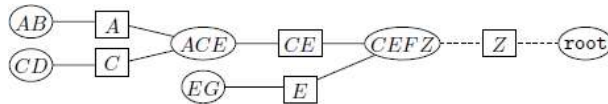
Orientation sans nouvelle V-structure

- Conclusion : avec 5000 cas, PC perd des informations sur des χ^2 faussés.



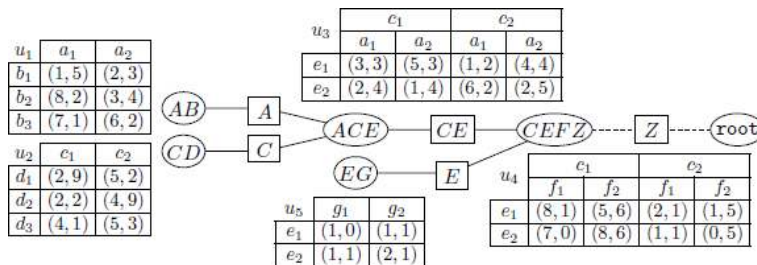
aGrUM : a library for decision
graphical models

Modèle graphiques



Formalisme permettant de manipuler :

- des lois de probabilité : réseaux bayésiens
- des utilités : réseaux GAI
- des contraintes : CSP



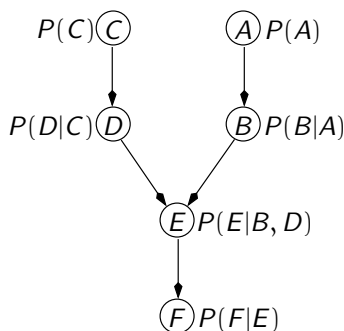
OpenTURNS'18

aGrUM

Modèle graphiques

13 / 57

Motivations pour aGrUM



- gestion des graphes
(orienté, non-orienté, mixte)
- algorithme efficace sur les graphes
(e.g., triangulations)
- gestion des tenseurs nommés $M(X_1, \dots, X_n)$
(quelque soit le type des paramètres)
- algorithme efficace sur les tenseurs nommés
(e.g., produit, projection, normalisation, etc.)

OpenTURNS'18

aGrUM

motivation

14 / 57

librairie aGrUM

aGrUM

- aGrUM = a Graphical Universal Modeler
- aGrUM = librairie C++ pour des applications incluant des modèles graphiques
- License : (L)GPL
- publique : <http://agrum.gitlab.io>
- cross-plateforme ([linux](#)/mac/windows)

- structures de données de base adaptées et sécurisées
- gestion légère et efficace des graphes
- gestion efficace des tenseurs nommés
- implémentation générique mais aussi implémentation efficace
- modèle graphique mais aucune interaction/dépendance avec une GUI
- Outils d'aide à la recherche
(e.g. générateurs aléatoire uniforme de BNs, introspection des classes, etc.)
- mais aussi outils d'aide à l'intégration
(e.g. *listener*, **pyAgnum**, multiples formats de fichiers, etc.)

OpenTURNS'18


aGrUM

librairie aGrUM

15 / 57

aGrUM pour les BNs

- Plusieurs types de variables aléatoires discrètes (labellisée, range, discrétisée)
- Plusieurs type de CPTs (array, parse, noisyOR, LogIt, aggregator, etc.)
- Plusieurs types d'inférences probabilistes
 - Shafer-Shenoy
 - Value Elimination
 - Lazy propagation
 - sampling (MonteCarlo, Gibbs, Importance, Weighted)
 - Loopy Belief Propagation
 - hybride LBP+Sampling
 - Plusieurs type de triangulations / constructions de l'arbre de jonction, etc.
- Apprentissages structures / paramètres.
 - Multiples algorithmes (HillClimbing, TabuSearch, K2, MIIC, 3off2),
 - multiples scores (LL, AIC, BIC, MDL, BDEU, BD, etc.),
 - multiples priors (NoPrior, Dirichlet, Laplace),
 - multiples contraintes (nb de parents, structure initiale, arc obligatoire, etc.)

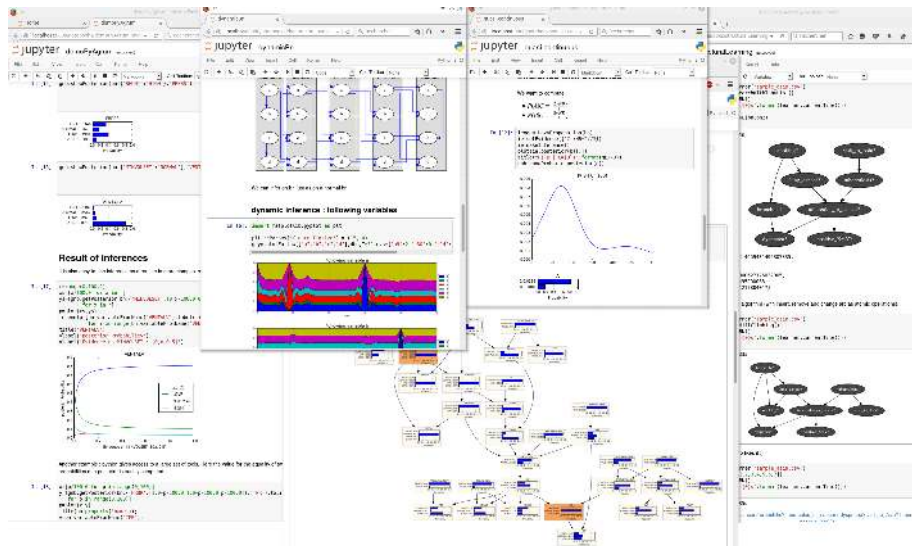
OpenTURNS'18 

aGrUM

librairie aGrUM

16 / 57

pyAgrum et iPython notebook



OpenTURNS'18 

aGrUM

librairie aGrUM

17 / 57

Nancy - le problème



- Base de 30 000 transactions immobilières ;
- Question : explication du prix immobiliers à partir des différents facteurs connus dans la base ;
- vers un outil de simulation d'impacts de décisions d'urbanisation.

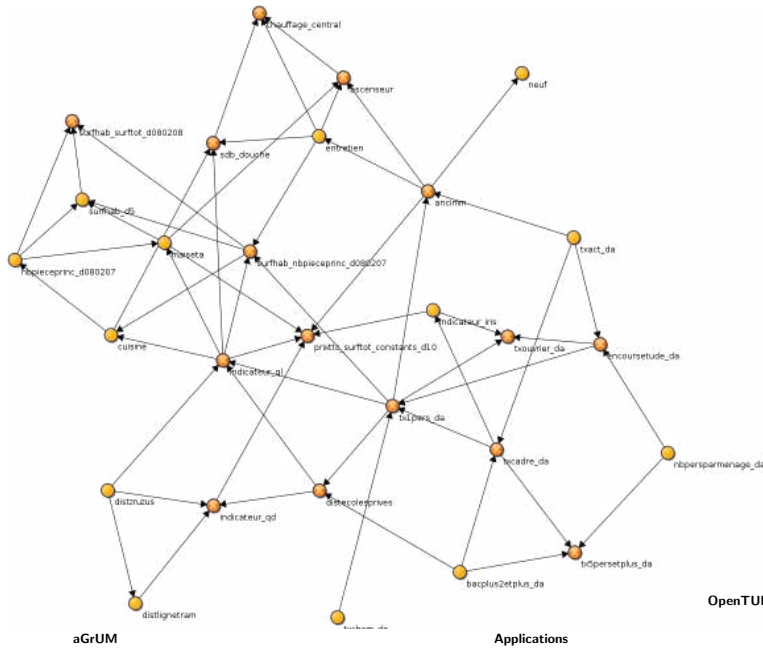
OpenTURNS'18 

aGrUM

Applications

18 / 57

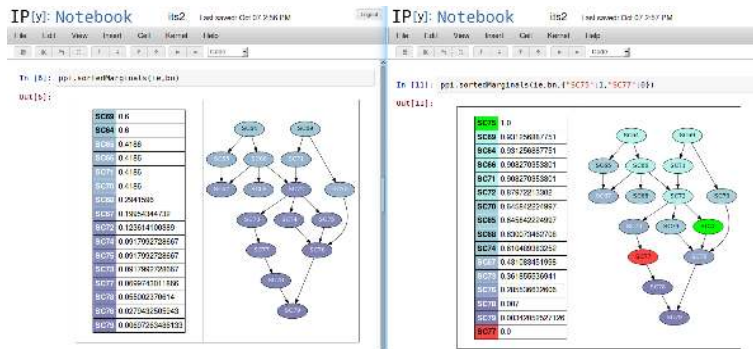
Nancy - le modèle



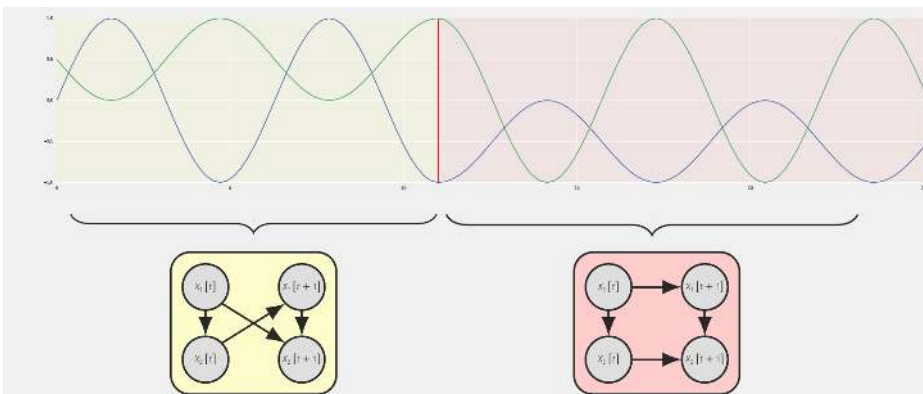
ITS - Intelligent Tutoring Systems

ITS : Network of skills

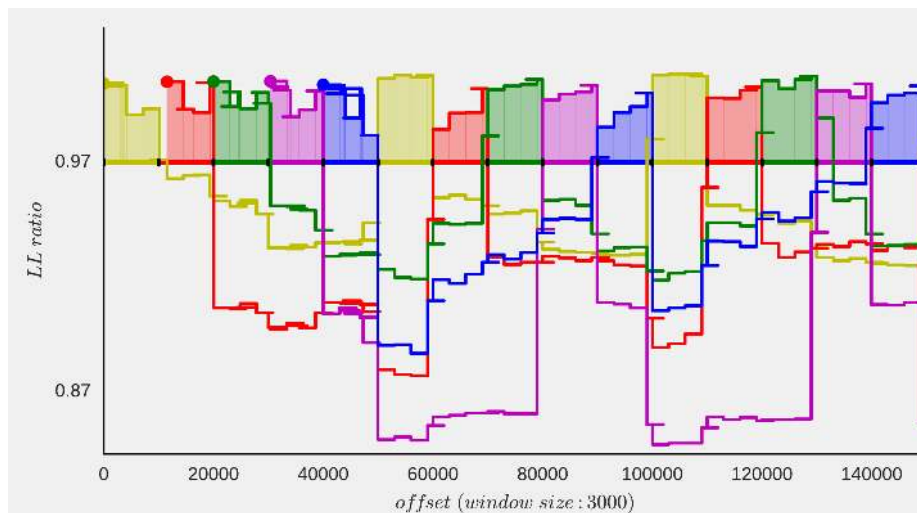
A network of skills explains which skills are necessary in order to learn a new one. This can be modeled as a BN with noisyAND nodes. Find the good next exercise is then an automatic diagnostic task (troubleshooting).



Apprentissage online et détection d'anomalies



- Streamed data, non i.i.d.
- Non homogène
- Modèle probabiliste dynamique
- Identification DBN et apprentissage incrémental



module otagr : Couplage openTURNS-aGrUM

- Né de la volonté de se parler discrètement et continuellement.
- Le point d'articulation principal : les distributions (jointes)

Communication aGrUM ↔ openTURNS

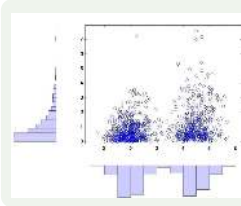
- `otagr.discretize()` depuis une marginale openTURNS vers un Potential (à étendre aux jointes. cf. exemple)
- `otagr.fromPotential()` et `otagr.fromMarginal()` depuis un Potential vers une `MixedHistogramUserDefined`



Demo !

Promesses d'un modèle graphique

- une *connaissance qualitative* sur le processus dont sont issus les données,
 - Validation,
 - Prédiction,
 - Explicabilité, etc.
- une *représentation compacte* : gain en temps et en mémoire (**inférence exacte, inférence approchée, sampling**),
- Une *approche causale*



Modèles graphiques et copules

Copules

Soit $X = \{X_1, \dots, X_n\}$ un ensemble de variables continues, sa CDF $F(x) = P(X \leq x)$, sa densité $p(x)$ liés par : $p(x) = \frac{\partial^n F}{\partial x_1 \dots \partial x_n}(x_1, \dots, x_n)$.

Copule

$C(u_1, \dots, u_n) =$ CDF de variables U_1, \dots, U_n distribuées uniformément sur $[0, 1]$

Fréchet-Hoeffding $\max(1 - n + \sum u_i, 0) \leq C(u) \leq \min(u_1, \dots, u_n)$

Sklar, 1959 $\forall F, \exists C$ copule s.t. $F(x_1, \dots, x_n) = C(F_1(x_1), \dots, F_n(x_n))$:
 $C(u) = F(F_1^{-1}(u_1), \dots, F_n^{-1}(u_n))$.

Unicité si continuité et $p > 0$

En densités, $p(x) = c(F_1(x_1), \dots, F_n(x_n)) \prod_{i=1}^n p_i(x_i)$.

Copule marginale et conditionnelle

$$C_k(u_1, \dots, u_k) = C(u_1, \dots, u_k, 1, \dots, 1) \quad C_k(u_k | u_1, \dots, u_{k-1}) = \frac{\frac{\partial^{k-1}}{\partial x_1 \dots \partial x_{k-1}} C_k}{\frac{\partial^{k-1}}{\partial x_1 \dots \partial x_{k-1}} C_{k-1}}$$

OpenTURNS'18



Mesure des dépendances et copules

Entropie

$$\mathcal{I}(X_1, X_2) = \iint_{u_1, u_2} c(u) \log c(u) du = -H(c).$$

Spearman's Rho (coefficient de corrélation des rangs)

$$\rho_S = \frac{\sum_m (R_X(m) - \overline{R_X})(R_Y(m) - \overline{R_Y})}{\sqrt{\sum_m (R_X(m) - \overline{R_X})^2 \sum_m (R_Y(m) - \overline{R_Y})^2}} \rightarrow 12 \iint C(x, y) dx dy - 3$$

C extrait la complexité structurelle des relations entre variables et s'abstrait de leurs comportements marginaux.



Copule $C(X)$ plus difficile à manipuler quand la dimension grandit.



Vine : Pair-Copula Construction

- Décomposition classique :

$$p(X_1, \dots, X_n) = p(X_1) \cdot p(X_2|X_1) \cdot p(X_3|X_1, X_2) \cdots p(X_n|X_1, \dots, X_{n-1})$$
- Pair-copula (récursive décomposition) :

$$p(X_1, X_t|X_2, \dots, X_{t-1}) = p(X_t|X_1, \dots, X_{t-1}) \cdot p(X_1|X_2, \dots, X_{t-1})$$

$$= c(X_1, X_t|X_2, \dots, X_{t-1}) \cdot p(X_t|X_2, \dots, X_{t-1}) \cdot p(X_1|X_2, \dots, X_{t-1})$$

$$p(X_t|X_1, \dots, X_{t-1}) = c(X_1, X_t|X_2, \dots, X_{t-1}) \cdot p(X_t|X_2, \dots, X_{t-1})$$

PCC decomposition

$$p(X_1, \dots, X_n) = \prod_{i=1}^n p(X_i) \cdot \left[\prod_{j=1}^{n-1} \prod_{k=1}^{n-j} c(X_i, X_{i+j}|X_{i+1}, \dots, X_{i+j-1}) \right]$$

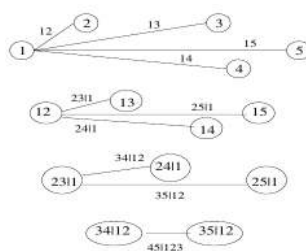
Cette décomposition n'est pas unique. Bedford and Cooke(2001) introduisent les *Regular Vine Tree Structure (RVTS)*.

OpenTURNS'18

C-Vine et D-Vine

C(anonical)-Vine

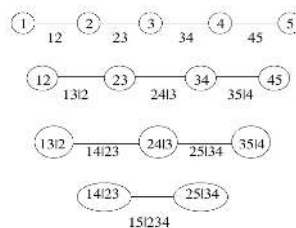
Chaque arbre de la RVTS a une unique nœud connecté à tous les autres.



$$C_{12345} = C_{12} \cdot C_{13} \cdot C_{14} \cdot C_{15} \cdot C_{23|1} \cdot C_{25|1} \cdot C_{24|1} \cdot C_{34|12} \cdot C_{35|12} \cdot C_{45|123}$$

D(?) -Vine

Chaque arbre est un chemin.

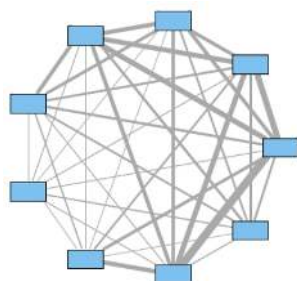


$$C_{12345} = C_{12} \cdot C_{23} \cdot C_{34} \cdot C_{45} \cdot C_{13|2} \cdot C_{24|3} \cdot C_{35|4} \cdot C_{14|23} \cdot C_{25|34} \cdot C_{15|234}$$

OpenTURNS'18

Décompositions de Vine

- Décomposition qui n'utilise pas explicitement une structure du modèle.
- Implicitement on choisit la RVTS en fonction d'a priori sur le modèle.
 - C-Vine : rangement des variables par importance
 - D-Vine : séquence de variables (causalité, temporalité, etc.)
- Des copules à haute dimension toujours ! (à moins de couper la décomposition en arbres avant la fin ?).
- Pas de simplification possible utilisant les indépendances (en fait si, on pourrait mais pas dans la littérature).
- Apprentissage possible : $n - 1$ maximum spanning-tree sur des corrélations entre couples de nœuds.



OpenTURNS'18

Cumulative Distribution Network

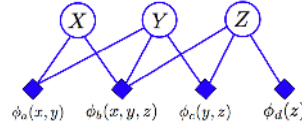
Soit $X = \{X_1, \dots, X_n\}$ et G un graphe non orienté sur X .

➡ Définition

CDN Un CDN est une distribution dont F se factorise sur les cliques de G :

$$\exists \{\phi_i\}_{S_i \text{ clique de } G} \text{ CDF sur } S_i, F(x_1, \dots, x_n) = \prod_{i=1}^m \phi_i(s_i)$$

- s_i est l'instanciation des variables de S_i dans x ;
- les S_i sont des cliques non nécessairement maximales ;
- un CDN factorise non pas p mais directement F ;
- une bonne représentation d'un CDN : le factor graph.



⚠ Les indépendances conditionnelles par un CDN sont très différentes des relations classiques dans un modèle graphique probabiliste (séparation, etc.)

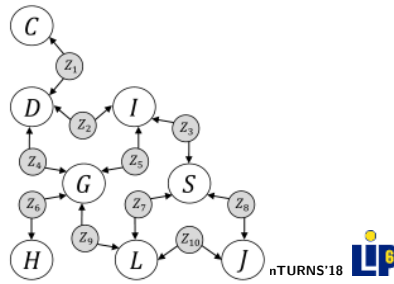
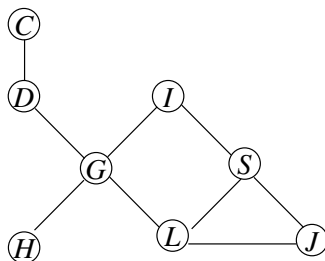
CDN, copules, indépendances conditionnelles

- Un CDN permet de factoriser des CDF mais pas des copules : un produit de copule n'est pas forcément une copule. Il faut modifier un petit peu les copules pour avoir le résultat attendu :

Soit $\{C_i\}_{S_i}$ famille de copules, soit d_i le nombre de clique qui contient X_i , alors

$$\text{Avec } v_i = u_i^{d_i}, C(X_1, \dots, X_n) = \prod_{i=1}^m C_i(v_i) \text{ est une copule sur } X$$

- Indépendances dans un CDN



Cumulative Distribution Network

- Factorisation directe d'une CDF sous la forme d'un produit de CDF sur des sous-ensembles de variables
- Représentation graphique : factor graph, graphe non orienté (CDF sur les cliques non forcément maximales)
- Pas de factorisation directe de copule : il faut une manipulation sur les variables.
- Indépendances particulières et peu intuitives (nécessité de variables latentes).

Les CDN sont un modèle graphique original mais qui n'a pas les avantages en modélisation, apprentissage et inférence d'un réseau bayésien.

Copula Bayesian Networks

Soit un réseau bayésien sur les densités de probabilités :

$$p(X_1, \dots, X_n) = \prod_{i=1}^n p(X_i | \pi_i)$$

Soit c copule pour p , soit $R_c(X_i, \pi_i) = \frac{c(F(X_i), F(P_1), \dots, F(P_{K_i}))}{\frac{\partial^{K_i}}{\partial F(P_1) \dots \partial F(P_{K_i})} c(1, F(P_1), \dots, F(P_{K_i}))}$

où k_i est le nombre de parents de X_i (si $k_i = 0$, $R_c(X_i, \pi_i) = 1$).

$$p(X_1, \dots, X_n) = \prod_{i=1}^n R_{c_i}(X_i, \pi_i) \cdot p(X_i)$$

Réciproquement, si les $\{c_i\}_{i=1}^n$ sont des copules (> 0) associées à chaque (X_i, π_i) avec π_i non vide, alors l'équation ci-dessus définit une copule sur X .

$$c(F(X_1), \dots, F(X_n)) = \prod_{i=1}^n R_{c_i}(X_i, \pi_i)$$

Copula Bayesian Network - Inférence et apprentissage

► Définition (Copula Bayesian Network)

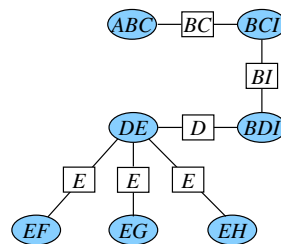
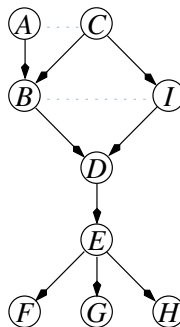
Un CBN est un triplet (G, Θ_C, Θ_p) où G est un DAG sur X , Θ_C est un ensemble de copule $c(X_i, \pi_i)$, Θ_p est un ensemble de marginale $p(X_i)$. Le CBN encode alors une densité de probabilité $p(X)$ factorisée par :

$$p(X_1, \dots, X_n) = \prod_{i=1}^n R_{c_i}(X_i, \pi_i) \cdot p(X_i)$$

Les même propriétés de localisation de l'information sont utilisable ici que sur un BN classique.

- La diminution de paramètres est souvent importante.
- L'apprentissage de paramètres se fait localement sur une base contenant X_i, π_i
 - possiblement pas la même base pour tous les R_c .
 - possiblement pas la même famille de copules pour chaque X_i, π_i
- L'apprentissage de structure peut utiliser le même genre d'algorithme que pour les BNs classiques (à base de score, à base de test).
- L'inférence est facilitée également : simulation d'une copule, transformation de Rosenblatt (etc) sont simplifiée!

Copula Bayesian Network – jointes au lieu de conditionnelles



$$p(X) = \frac{\prod_{C \in \text{clique}(G)} p(C)}{\prod_{S \in \text{separateur}(G)} p(S)}$$

Soit $c_S(S)$ des copules sur les ensembles $S \in X$ (clique ou séparateur, avec certaines propriétés de cohérence)

$$c(X) = \frac{\prod_{C \in \text{clique}(G)} c_C(C)}{\prod_{S \in \text{separateur}(G)} c_S(S)}$$

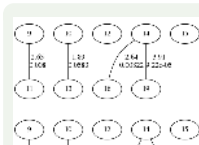
est une copule pour $p(X)$

Apprentissage de structure dans les CBN

- Algorithme glouton basé sur des scores : estimation de la log-vraisemblance d'une structure + pénalisation de la complexité du modèle (BIC, AIC, etc.)
- Algorithme (rapide) utilisant le coefficient de Spearman comme "proxy" de la vraisemblance.
- Mais pas de méthode, en particulier pour gérer le compromis entre le "proxy" et la pénalité de la complexité du modèle.
- Problème de la qualité asymptotique de la relation entre Spearman et vraisemblance : mauvais prise en compte des bases de taille modeste ?

Test d'indépendance non paramétrique

Il y a un intérêt à trouver une méthode robuste de test des indépendances conditionnelles dans une base de taille modeste.



Apprentissage structurel Modèles graphiques dans les copules

Apprentissage structurel pour les modèles graphiques de copules

Test d'indépendance non paramétrique : $Y \perp\!\!\!\perp Z | X$

- Le test a été proposé par Bouezmarni et al, 2009.
- Il est basé sur la distance de Hellinger entre la copule jointe et le produit des copules marginales,

$$H = \int_{[0,1]^{d+2}} \left(1 - \sqrt{\frac{c_{XY}(\mathbf{x}, y) c_{XZ}(\mathbf{x}, z)}{c_{XYZ}(\mathbf{x}, y, z)}} \right)^2 c_{XYZ}(\mathbf{x}, y, z) d\mathbf{x} dy dz,$$

- Toute copule est approchée par une copule de Bernstein (estimation non paramétrique) sur une base de N échantillons :

$$\hat{H} \approx \frac{1}{N} \sum_{i=1}^N \left(1 - \sqrt{\frac{\hat{c}_{XY}(\bar{F}_X(\mathbf{x}_i), F_Y(y_i)) \hat{c}_{XZ}(\bar{F}_X(\mathbf{x}_i), F_Z(z_i))}{\hat{c}_{XYZ}(\bar{F}_X(\mathbf{x}_i), F_Y(y_i), F_Z(z_i))}} \right)^2.$$

$Y \perp\!\!\!\perp Z | X ?$

[Bouezmarni et al, 2009] montre que sous $H_0 : [Y \perp\!\!\!\perp Z | X], T \sim \mathcal{N}(0, 1)$:

$$T \equiv \frac{Nk^{-(d+2)/2}}{\sigma} (4H - N^{-1}C_1k^{(d+2)/2} - N^{-1}B_1k^{(d+1)/2} - N^{-1}B_2k^{(d+1)/2} - N^{-1}B_3k^{d/2})$$

avec k fixé arbitrairement à \sqrt{d} et :

$$C_1 = 2^{-(d+2)}\pi^{(d+2)/2}, \quad \sigma = \sqrt{2}(\pi/4)^{(d+2)/2},$$

$$B_1 = -2^{-d}\pi^{(d+1)/2} + \frac{1}{N} \sum_{i=1}^N \frac{\prod_{j=1}^{d+1} (4\pi g_j^{(i)} (1 - g_j^{(i)}))^{-1/2}}{c_{XY}(g_1^{(i)}, \dots, g_{d+1}^{(i)})},$$

$$B_2 = -2^{-d}\pi^{(d+1)/2} + \frac{1}{N} \sum_{i=1}^N \frac{4\pi (g_{d+2}^{(i)} (1 - g_{d+2}^{(i)}))^{-1/2} \prod_{j=1}^d (4\pi g_j^{(i)} (1 - g_j^{(i)}))^{-1/2}}{c_{XZ}(g_1^{(i)}, \dots, g_d^{(i)}, g_{d+2}^{(i)})},$$

$$B_3 = 2^{-(d+1)}\pi^{-d/2} \frac{1}{N} \sum_{i=1}^N \frac{c_X(g_1^{(i)}, \dots, g_d^{(i)})}{\sqrt{\prod_{j=1}^d g_j^{(i)} (1 - g_j^{(i)})}},$$

Algorithme PC proposé

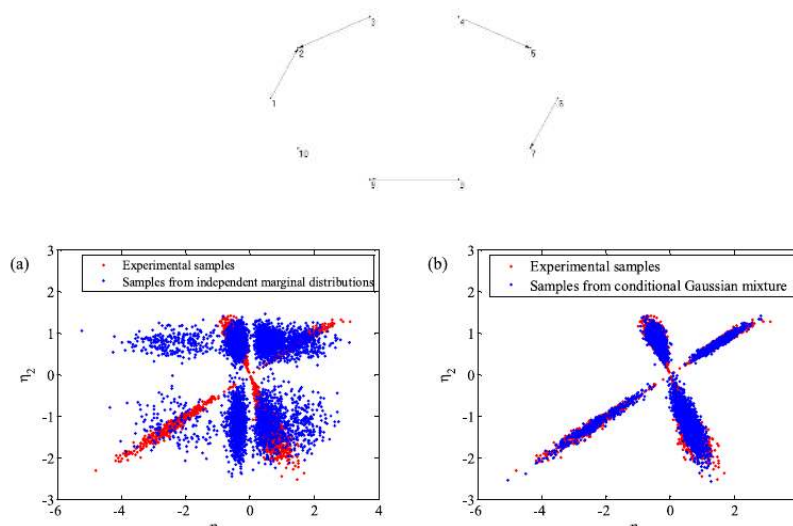
Algorithm 1 Construction of an undirected graphical model using CI tests.

```

• Start with a complete undirected graph  $G = (V, E)$  where  $V$  is the node set and  $E$  is the edge set.
• Set conditional independence test order  $n = 0$ .
repeat
  for  $Y \in V$  do
    for  $Z \in \text{Adjacencies}(Y)$  do
      for  $S \subseteq \text{Adjacencies}(Y) \setminus \{Z\}$  and  $|S| = n$  do
        if  $Y \perp\!\!\!\perp Z | S$  then
          remove edge that connects  $Y$  and  $Z$  from  $E$ , and update the undirected graph  $G$ .
        end if
      end for
    end for
  end for
   $n = n + 1$ 
until  $|\text{Adjacencies}(Y) \setminus \{Z\}| < n$  or  $n = n_{\max}$ 

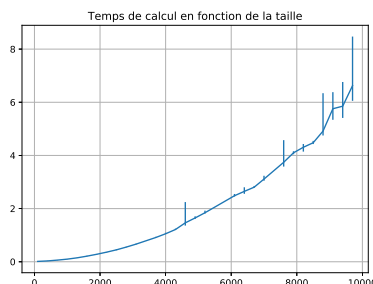
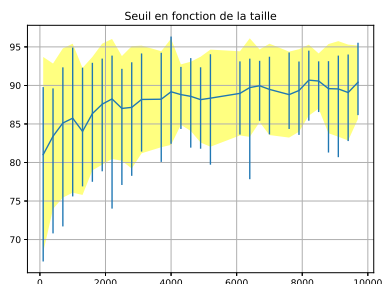
```

Résultats issus de la littérature

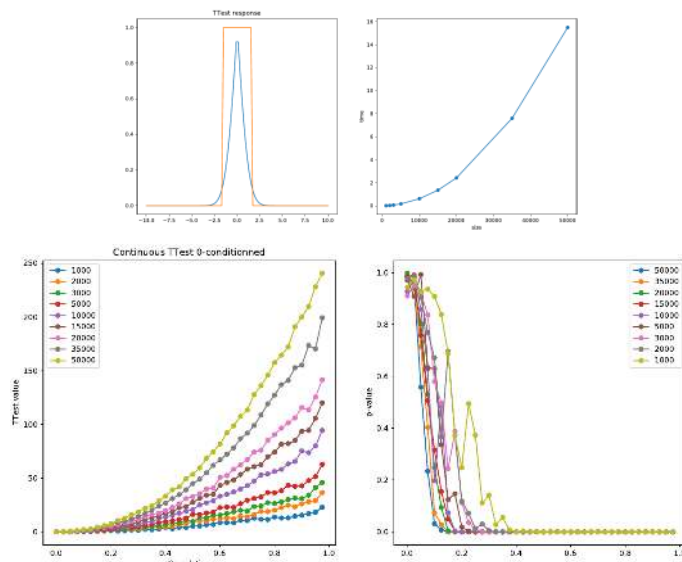


Implémentation du test

- Ce test a été implémenté en C++/openTurn.
- Accessible en python dans un module otagr (OpenTurns/aGrUM).
- Calcul de la p value pour vérifier la normalité de T .
- Pour tester ce test,
 - Tout test de 5 variables indépendantes (donc 10 tests),
 - On garde le seuil en p value minimum pour reconnaître toutes les indépendances,
 - Pour chaque N , 20 expériences



Test du test sur variables gaussiennes corrélées



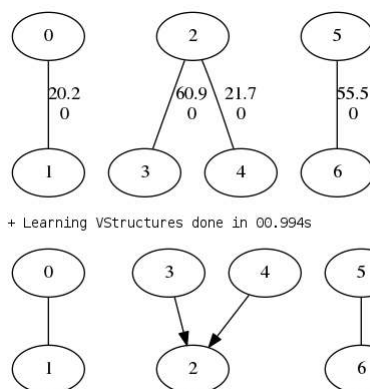
Implémentation de l'algorithme continuous-PC

- Implémentation finalisée.
- Mise en place d'un cache "adaptatif" : les mêmes copules peuvent être utilisés plusieurs fois mais (sur 3 itérations consécutives).
- Temps de calcul très variable.
- Test de la pertinence difficile à faire pour l'instant : il faut pouvoir générer des CBN afin de les sampler et d'essayer de les apprendre.
- Orientation des arcs : la version proposée par l'article n'est pas acceptable (empirique) \Rightarrow Utilisation du test d'indépendance classique pour PC.
- Utilisation de aGrUM pour l'apprentissage, l'analyse du graphe en arbre de jonction et la visualisation.

continuous-PC : premier test

3 copules différentes mises en parallèle dans une copule jointe.

```
def generateDataForSpecificInstance(20000, iteration=3, alpha=0.1):
    with tqdm(desc="Generating data") as pbar:
        for i in range(iteration):
            # Generating data
            data = generateDataForSpecificInstance(20000, iteration=i, alpha=alpha)
            # Learning Vstructures
            vstructures = learnVstructures(data, alpha=alpha)
            # Evaluating the Vstructures
            score = evaluateVstructures(vstructures, data)
            pbar.set_postfix(score=score)
            # Printing the Vstructures
            printVstructures(vstructures)
    return vstructures
```

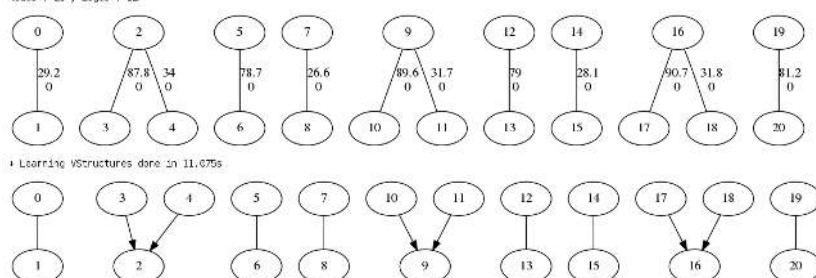


+ Learning VStructures done in 00.994s

continuous-PC : premier test

3 même copules différentes mises en parallèle répétées plusieurs fois.

```
1 | learnGenerateDataForSpecificInstance(20000, iteration=3, alpha=0.1)
executed in 00.54s, finished 18:03:08 2018-09-27
+ Generating done in 00.138s
Sample : 20000x21
+ Initializing done in 00.026s
+ Learning skeleton done in 03:42.040s
Nodes : 21, Edges : 12
```

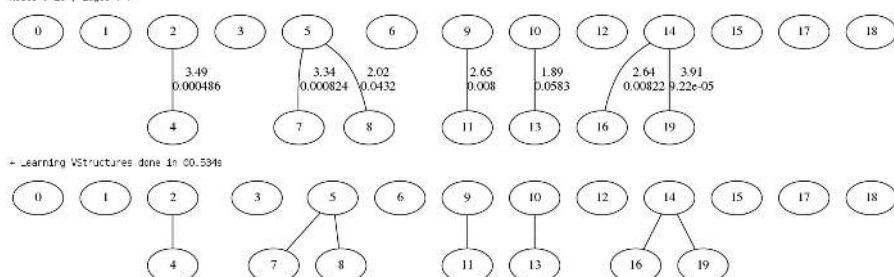


+ Learning VStructures done in 11.675s

continuous-PC : second test (effet de size)

Des copules jointes en blocs de tailles croissantes.

```
+ Reading BlockCopula/BlockCopula_increasing_blocks_dimension_20_size_10000.csv done in 00.152s
Sample : 5000x20
+ Initializing done in 00.005s
+ Learning skeleton done in 12.845s
Nodes : 20, Edges : 7
```

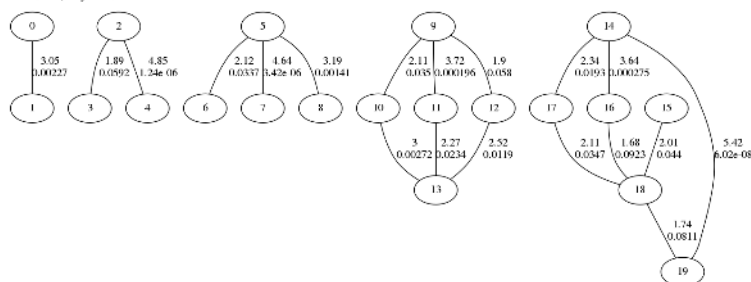


+ Learning VStructures done in 00.534s

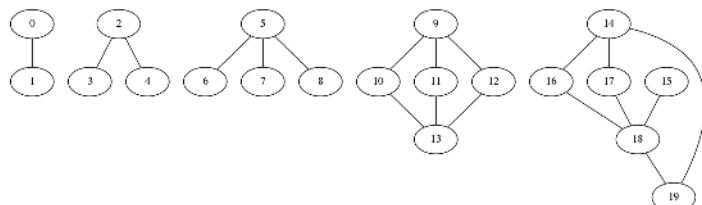
continuous-PC : second test (effet de size)

Des copules jointes en blocs de tailles croissantes.

- Loading: 0.000000/0.000000, increasing_blocks_constructor_02_01n_10000.csv done in 00.194s
 sample = 1000000
 - Initializing done in 00.011s
 - Learning: 0.000000 done in 45.106s
 Nodes : 20, Edges : 15

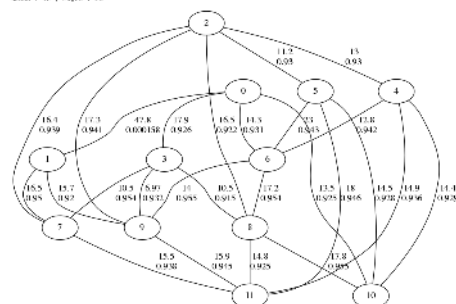
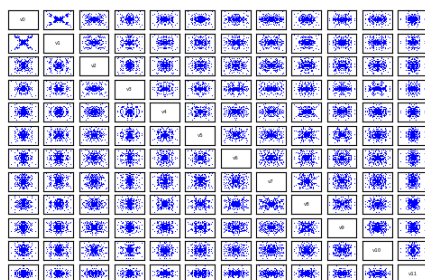


- Learning VStructures done in 10.47%

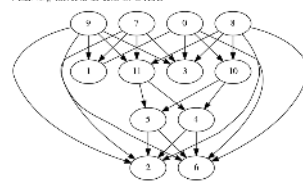


continuous-PC : Copules complexes

- Loading: 0.000000/0.000000
 - Initializing done in 00.000s
 - Learning: 0.000000 done in 10.47405s
 Nodes : 12, Edges : 12

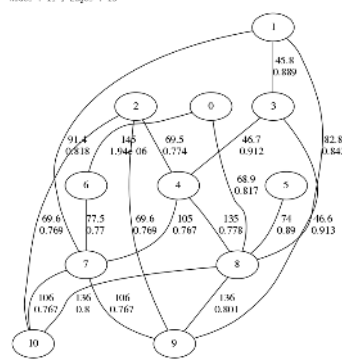
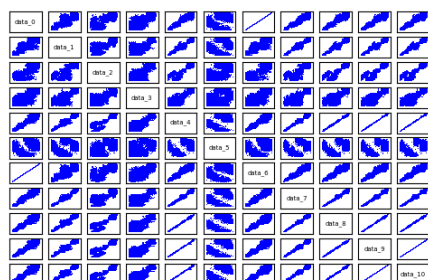


- Learning VStructures done in 37.804s

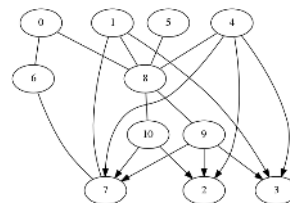


continuous-PC : cas réel

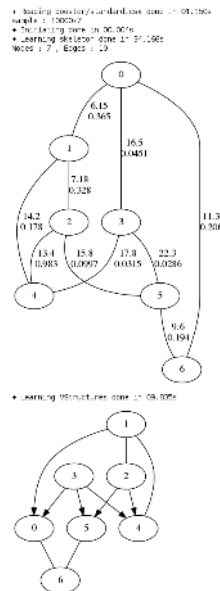
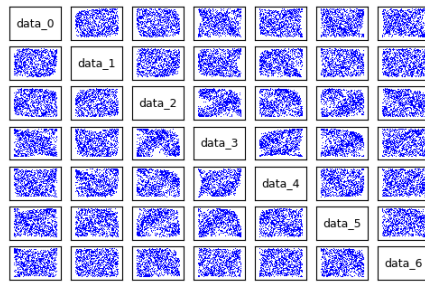
- Initializing done in 03.006s
 - Learning: 0.000000 done in 00.30.270s
 Nodes : 11, Edges : 10



- Learning VStructures done in 21.528s

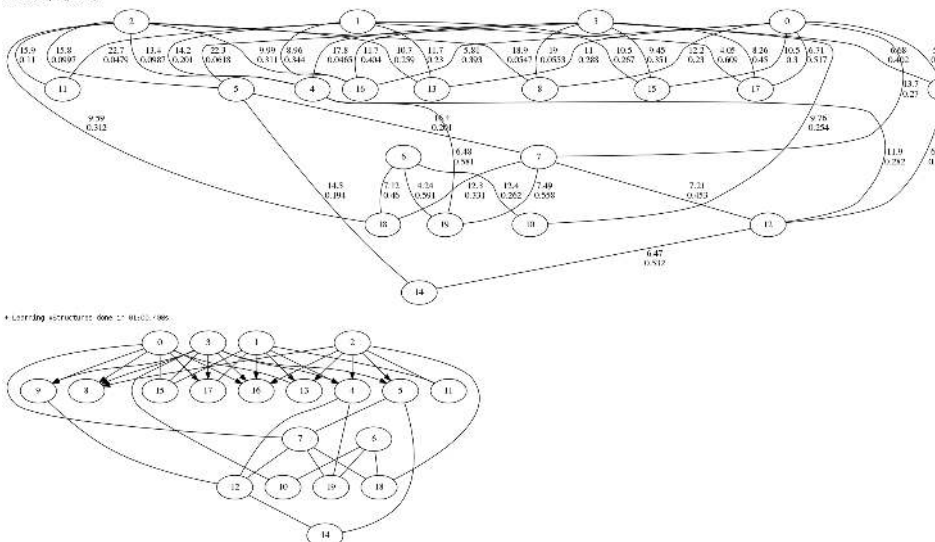


continuous-PC : transformation KL (taille 7)



continuous-PC : transformation KL (taille 20)

* Sampling structure: continuous data: 0.000000
 sample: 1000000
 * The learning time: 0.000000
 * Learning structure: done at 0.000000
 Nodes: 20, Edges: 40



Conclusions

- Plusieurs modèles graphiques différents pour copules
- Le "bon" modèle = CBN. Mais une littérature peu nombreuse. Certainement lié à des difficultés techniques ?










Certainement une intersection entre modèles graphiques et copules intéressante et encore à explorer en grande partie.

- Représentabilité en grande dimensions,
- Optimisation des inférences et simulations,










Collaboration OpenTurns-aGrUM plus poussée (et intriquée) que les premiers tests basée sur :

- ajout de la JunctionTreeCopula
- ajout du code d'apprentissage de la JunctionTreeCopula (ce qui implique la présence d'objet JunctionTree et de code d'apprentissage d'aGrUM accessible à partir d'OpenTurns).
- ajout du modèle CBN dans otagr

References I

-  Tim Bedford and Roger M. Cooke, *Probability density decomposition for conditionally dependent random variables modeled by vines.*, Ann. Math. Artif. Intell. **32** (2001), no. 1-4, 245–268.
-  Taoufik Bouezmarni, Jeroen Rombouts, and Abderrahim Taamouti, *A nonparametric copula based test for conditional independence with applications to granger causality*, Economics working papers, Universidad Carlos III, Departamento de Economía, 2009.
-  J. Dißmann, E. C. Brechmann, C. Czado, and D. Kurowicka, *Selecting and estimating regular vine copulae and application to financial returns*, Comput. Stat. Data Anal. **59** (2013), 52–69.
-  Gal Elidan, *Copula bayesian networks*, Advances in Neural Information Processing Systems 23 : 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada., 2010, pp. 559–567.
-  ———, *Inference-less density estimation using copula bayesian networks*, UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010, 2010, pp. 151–159.
-  ———, *Lightning-speed structure learning of nonlinear continuous networks*, Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012, 2012, pp. 355–363.
-  Jim C. Huang and Brendan J. Frey, *Cumulative distribution networks and the derivative-sum-product algorithm : Models and inference for cumulative distribution functions on graphs.*, Journal of Machine Learning Research **12** (2011), 301–348.

References II

-  Jim C. Huang and Nebojsa Jojic, *Maximum-likelihood learning of cumulative distribution functions on graphs*, 13th International Conference on Artificial Intelligence and Statistics, AISTATS, 2010.
-  Jim C Huang, *Cumulative distribution networks : Inference, estimation and applications of graphical models for cumulative distribution functions*, Ph.D. thesis, University of Toronto, 2009.
-  Nebojsa Jojic, Chris Meek, and Jim C. Huang, *Exact inference and learning for cumulative distribution functions on loopy graphs*, Advances in Neural Information Processing Systems 23 (J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, eds.), Curran Associates, Inc., 2010, pp. 874–882.
-  Ricardo Silva, *Bayesian inference in cumulative distribution fields*, Interdisciplinary Bayesian Statistics (Adriano Polpo, Francisco Louzada, Laura L. R. Rifo, Julio M. Stern, and Marcelo Lauretto, eds.), Springer Proceedings in Mathematics & Statistics, vol. 118, Springer International Publishing, 2015, pp. 83–95.
-  Yaniv Tenzer and Gal Elidan,  *HELM : highly efficient learning of mixed copula networks*, Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27, 2014, 2014, pp. 790–799.
-  Stefan Douglas Webb, *Inference, sampling, and learning in copula cumulative distribution networks.*, CoRR abs/1310.4456 (2013).
-  Jiang Wan and Nicholas Zabaras,  *a probabilistic graphical model based stochastic input model construction*, J. Comput. Physics **272** (2014), 664–685.