



Probabilistic Graphical Models, pyAgrum & Causality

Pierre-Henri WUILLEMIN & Gaspard DUCAMP

LIP6 & ex-LIP6

{17|22}/06/2021



Introduction

Réseaux bayésiens (discrets)

Definition

Inference

Applications

Use Cases

Dynamic Bayesian Networks

aGrUM/pyAgrum

Learning Bayesian Networks



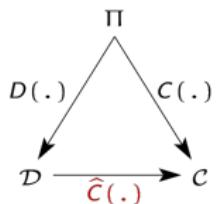
DISCLAIMER



- ▶ Niveaux de connaissances
- ▶ Hypothèse : classification \in ppcc
- ▶ Classification versus BN
- ▶ ! Franglish in transparents

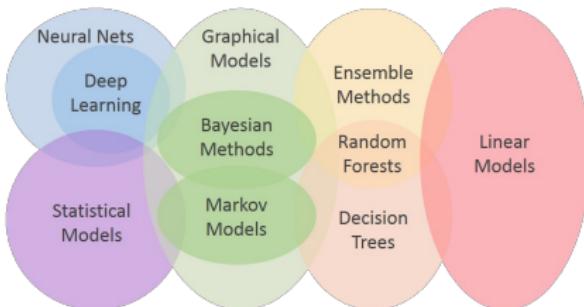
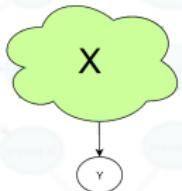


Classification in one slide

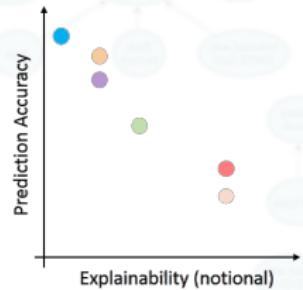


Classification (probabiliste)

- ▶ $\hat{y} = \hat{C}(x) \approx C(D^{-1}(x)) = f(x)$
- ▶ $P(x|y) = f(x)$
 - ▶ $\hat{y}_{ML} = \hat{C}(x) = \arg \max_y P(x|y)$
 - ▶ $\hat{y}_{MAP} = \hat{C}(x) = \arg \max_y P(y|x)$



Source : Data Science Lab - SIA partners



Paradoxe de Simpson - data



Patient	Drug	Gender	
0	Sick	With	F
1	Sick	Without	M
2	Healed	Without	M
3	Healed	With	F
4	Sick	With	M
...
1495	Healed	Without	M
1496	Healed	With	F
1497	Sick	With	F
1498	Healed	Without	F
1499	Sick	With	F

1500 rows × 3 columns

		Gender	
Patient	Drug	F	M
Healed	With	396	29
	Without	153	218
Sick	With	185	139
	Without	43	337

```
4 # pyAgrum
5 learner=gum.BNLearner("simpson.csv")
6 learner.pseudoCount(["Gender", "Drug", "Patient"])
```

Paradoxe de Simpson - calculs



		Patient	
Drug	Gender	Healed	Sick
With	F	0.2460	0.1093
	M	0.0200	0.1013
Without	F	0.1187	0.0220
	M	0.1540	0.2287

$P(Patient, Gender, Drug)$

```
p.normalize()
```

		Patient	
Drug	Healed	Sick	
With	0.5580	0.4420	
Without	0.5210	0.4790	

$P(Patient|Drug)$

```
p.margSumOut("Gender")/p.margSumIn("Drug")
```

$$\begin{aligned} 0.5580 &> 0.5210 \\ \Rightarrow \textit{With} &\succ \textit{Without} \end{aligned}$$

		Patient	
Drug	Gender	Healed	Sick
With	F	0.6923	0.3077
	M	0.1648	0.8352
Without	F	0.8436	0.1564
	M	0.4024	0.5976

$P(Patient|Gender, Drug)$

```
p/p.margSumOut("Patient")
```

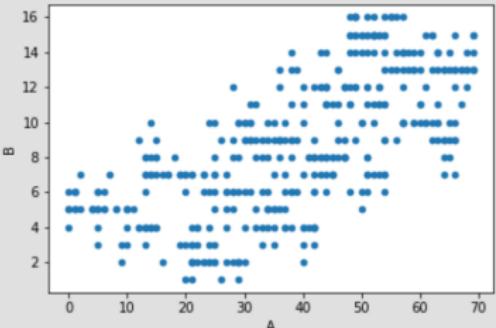
$$0.6923 < 0.8436$$

\Rightarrow sauf si F

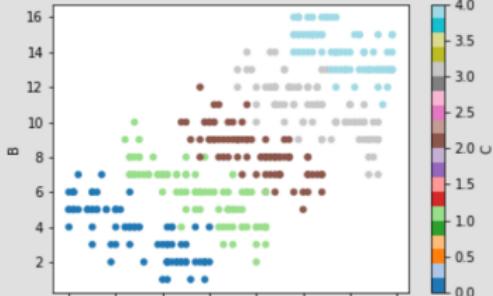
$$0.1648 < 0.4024$$

\Rightarrow et sauf si H

Paradoxe de Simpson - autre version



the trend is increasing



the trend is decreasing for any value for C !

Conclusions sur Simpson



Quoi ?



Comment ?



Et donc ?





Introduction

Réseaux bayésiens (discrets)

Definition

Inference

Applications

Use Cases

Dynamic Bayesian Networks

aGrUM/pyAgrum

Learning Bayesian Networks





Joint Probabilistic Model

With A, C discrete random variables,

- ▶ Joint Probability : $P(A, C)$
- ▶ Marginal Probability : $P(A) = \sum_C P(A, C)$
- ▶ Conditional Probability : $P(C|A) = \frac{P(A, C)}{P(A)}$

Inference

Inference consists in computing the distribution of one variable (C) given observations on some of the others (A).

$$P(C|A = \epsilon_a) = P(C|\epsilon_a) = \frac{P(\epsilon_a|C) \cdot P(C)}{P(\epsilon_a)} = \frac{P(C, \epsilon_a)}{P(\epsilon_a)} \propto P(C, \epsilon_a)$$



Inference

Inference consists in computing the distribution of one variable (C) given observations on some of the others (A).

$$P(C|A = \epsilon_a) = P(C|\epsilon_a) = \frac{P(\epsilon_a|C) \cdot P(C)}{P(\epsilon_a)} = \frac{P(C, \epsilon_a)}{P(\epsilon_a)} \propto P(C, \epsilon_a)$$

		C	
		0	1
A	0	0.4235	0.0793
	1	0.3929	0.1043

C	
0	1
0.3929	0.1043
0.3929	0.1043

C	
0	1
0.7902	0.2098
0.7902	0.2098

$P(A, C)$

$P(A = 1, C)$

$P(C|A = 1)$



Probabilistic complex systems



$P(X_1, \dots, X_n) \Rightarrow O(k^n)$ in space

Inference in complex systems : $P(X_i|X_j = \epsilon_j) \propto P(X_i, \epsilon_j)$

$P(X_i|\epsilon_j) = \frac{P(X_i, \epsilon_j)}{P(\epsilon_j)} \propto \sum_{k \notin \{i,j\}} P(\dots, X_i, \dots, \epsilon_j, \dots) \Rightarrow O(k^n)$ in time

		D	
		B	C
A	0	0.3383	0.0566
	1	0.0227	0.0059
B	0	0.0511	0.0193
	1	0.0069	0.0002
C	0	0.1104	0.0185
	1	0.2098	0.0543
D	0	0.0167	0.0063
	1	0.0793	0.0020

C		
A	0	1
0	0.4235	0.0793
1	0.3929	0.1043

C		
	0	1
0	0.3929	0.1043
1	0.7902	0.2098

$$P(A, B, C, D)$$

$$P(A, C) = \sum_{B,D} P(A, B, C, D)$$

$$P(A = 1, C)$$

$$P(C | A = 1)$$



Combinatorial explosion, curse of dimensionality !



Bayesian Network – Model



Bayesian networks : definition

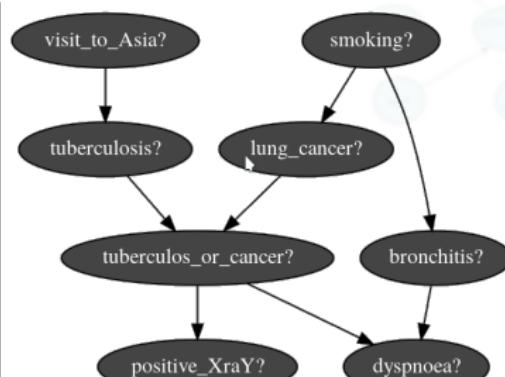
▶ Definition (Bayesian Network (BN))

A Bayesian network is a joint distribution over a set of random (discrete) variables.

A Bayesian network is represented by a directed acyclic graph (DAG) and by a conditional probability table (CPT) for each node X_i :

$$P(X_i|\text{parents}_i)$$

	lung_cancer?	
smoking?	e1	e2
f1	0.1000	0.9000
f2	0.0100	0.9900



	tuberculosis?	
visit_to_Asia?	b1	b2
a1	0.0500	0.9500
a2	0.0100	0.9900

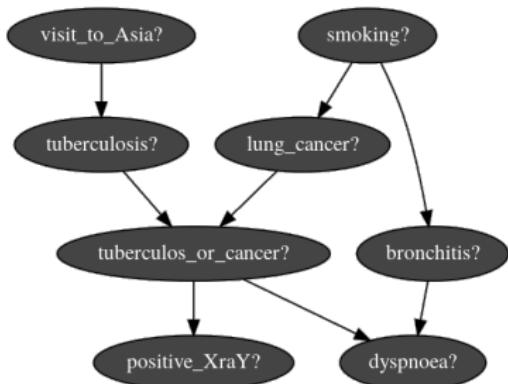


▶ Definition (Bayesian Network (BN))

A Bayesian network is a joint distribution over a set of random (discrete) variables.

A Bayesian network is represented by a directed acyclic graph (DAG) and by a conditional probability table (CPT) for each node

$$P(X_i|\text{parents}_i)$$



		dyspnoea?	
		h1	h2
c1	g1	0.9000	0.1000
	g2	0.7000	0.3000
c2	g1	0.8000	0.2000
	g2	0.1000	0.9000



▶ Definition (Bayesian Network (BN))

A Bayesian network is a joint distribution over a set of random (discrete) variables.

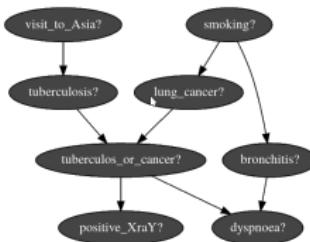
A Bayesian network is represented by a directed acyclic graph (DAG) and by a conditional probability table (CPT) for each node $P(X_i|\text{parents}_i)$

Factorization of the joint distribution in a BN

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i|\text{parents}(X_i))$$

$$P(A, S, T, L, O, B, X, D)$$

$$(2^8 = 256 \text{ parameters})$$



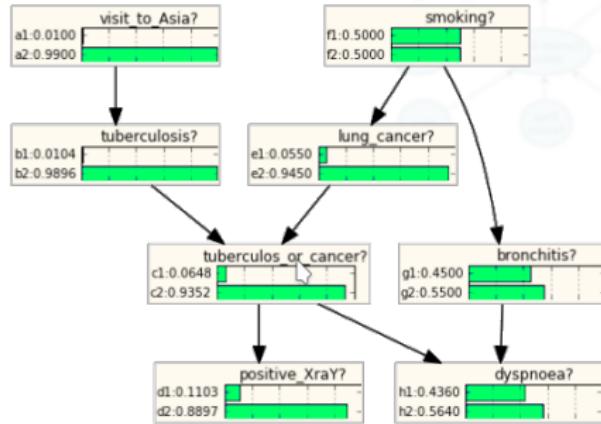
$$\begin{aligned} P(A) &\cdot P(S) \cdot P(T|A) \\ P(L|S) &\cdot P(O|T, L) \\ P(B|S) &\cdot P(X|O) \\ P(D|O, B) & \end{aligned}$$

$$\begin{aligned} (2+2+4+4+8+4+4+8 =) \\ 32 \text{ parameters} \end{aligned}$$

▶ Definition (Bayesian Network (BN))

A Bayesian network is represented by a directed acyclic graph (DAG) and by a conditional probability table (CPT) for each node
 $P(X_i|\text{parents}_i)$

Inference : $P(\text{dyspnoea}) ?$



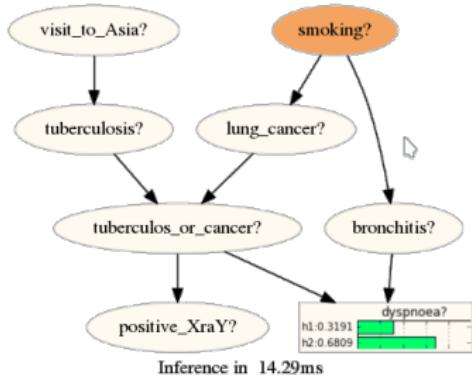
Bayesian networks : definition



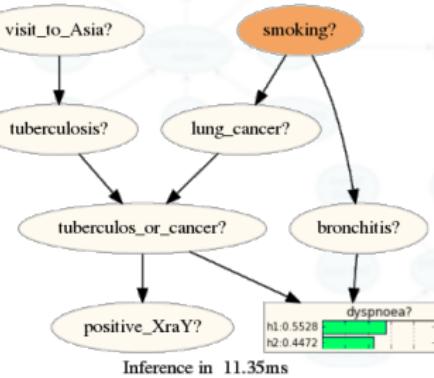
▶ Definition (Bayesian Network (BN))

A Bayesian network is represented by a directed acyclic graph (DAG) and by a conditional probability table (CPT) for each node
 $P(X_i|\text{parents}_i)$

Inference : $P(\text{dyspnoea}|\text{smoking})?$



$P(\text{dyspnoea}|\text{smoking} = 1)$



$P(\text{dyspnoea}|\text{smoking} = 0)$



exemple de la dyspnée (Lauritzen & Spiegelhalter (88))

- La **dyspnée** peut être engendrée par une **tuberculose**, un **cancer des poumons**, une **bronchite**, par plusieurs de ces maladies (... ou bien par autre chose).
- Un séjour récent en **Asie** augmente les chances de tuberculose, tandis que **fumer** augmente les risques de cancer des poumons.
- Des **rayons X** permettent de détecter une tuberculose ou un cancer.

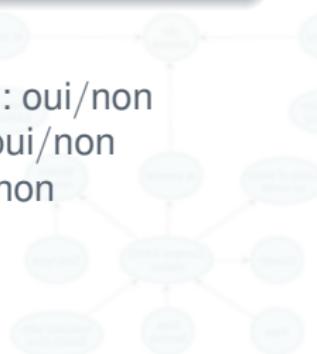
Variables and relations :

- D : dyspnée : oui/non
- C : cancer : oui/non
- A : Asie : oui/non
- R : rayons X : positif/négatif
- T : tuberculose : oui/non
- B : bronchite : oui/non
- F : fumer : oui/non



Probabilités :

- $P(T|A = \text{oui}) = ?$
- $P(T|A = \text{non}) = ?$
- $P(C|F) = ?$

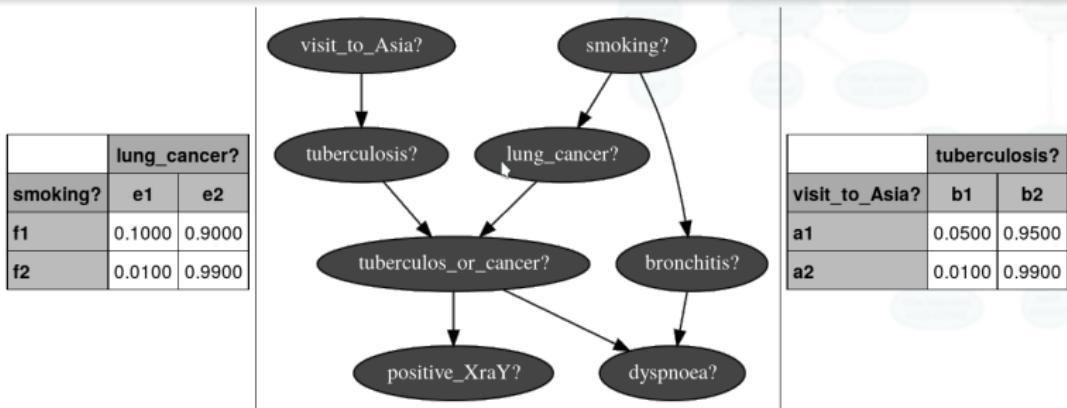


Un exemple



example de la dyspnée (Lauritzen & Spiegelhalter (88))

- La **dyspnée** peut être engendrée par une **tuberculose**, un **cancer des poumons**, une **bronchite**, par plusieurs de ces maladies (... ou bien par autre chose).
- Un séjour récent en **Asie** augmente les chances de tuberculose, tandis que **fumer** augmente les risques de cancer des poumons.
- Des **rayons X** permettent de détecter une tuberculose ou un cancer.

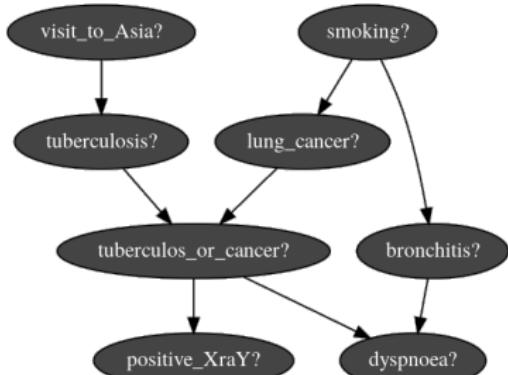


Un exemple



example de la dyspnée (Lauritzen & Spiegelhalter (88))

- La **dyspnée** peut être engendrée par une **tuberculose**, un **cancer des poumons**, une **bronchite**, par plusieurs de ces maladies (... ou bien par autre chose).
- Un séjour récent en **Asie** augmente les chances de tuberculose, tandis que **fumer** augmente les risques de cancer des poumons.
- Des **rayons X** permettent de détecter une tuberculose ou un cancer.



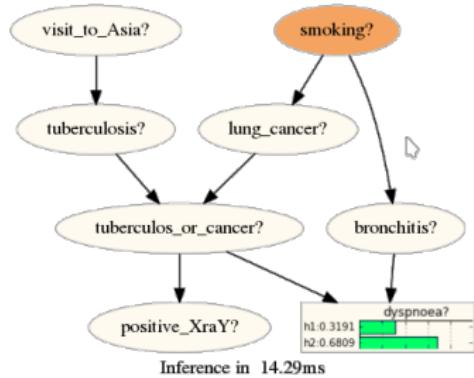
		dyspnoea?	
		h1	h2
c1	g1	0.9000	0.1000
	g2	0.7000	0.3000
c2	g1	0.8000	0.2000
	g2	0.1000	0.9000



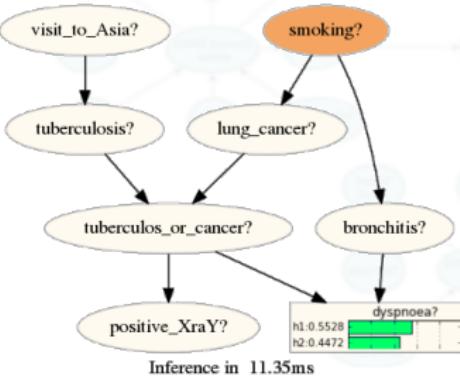
▶ Definition (Réseau bayésien (BN))

Un réseau bayésien est représenté par un graphe orienté sans circuit (DAG) and par des distribution de probabilités conditionnelles : $P(X_i|\text{parents}_i)$

Inférences probabilistes : $P(\text{dyspnée}|\text{smoking})$?



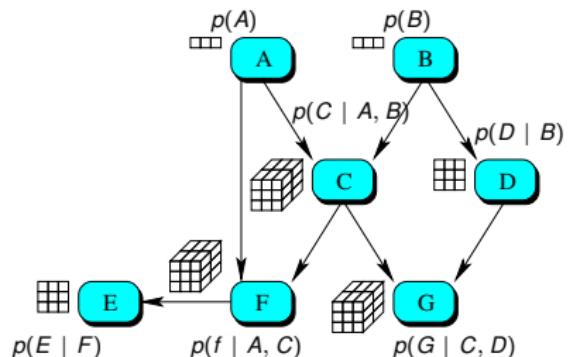
$$P(\text{dyspnoealsmoking} = 1)$$



$$P(\text{dyspnoealsmoking} = 0)$$



diagnostic : $P(A|F)$



- diagnostic

- reliability

- classification

prediction $P(E|B, A)$

- Process simulation (modélisation)

- forecasting (dynamics, etc.)

- Behavioral analysis (bot, intelligent tutoring system)

Others tasks

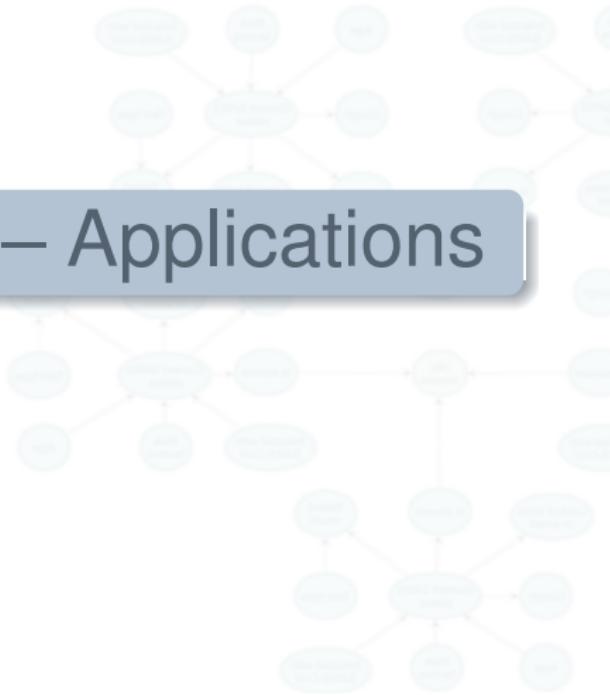
- Most Probable Case : $\arg \max P(\mathfrak{X}|D)$

- Sensitivity analysis, Informational analysis (mutual information), etc.

- Decision process, Troubleshooting : $\arg \max \frac{P(\cdot)}{C(\cdot)}$



Bayesian Network – Applications



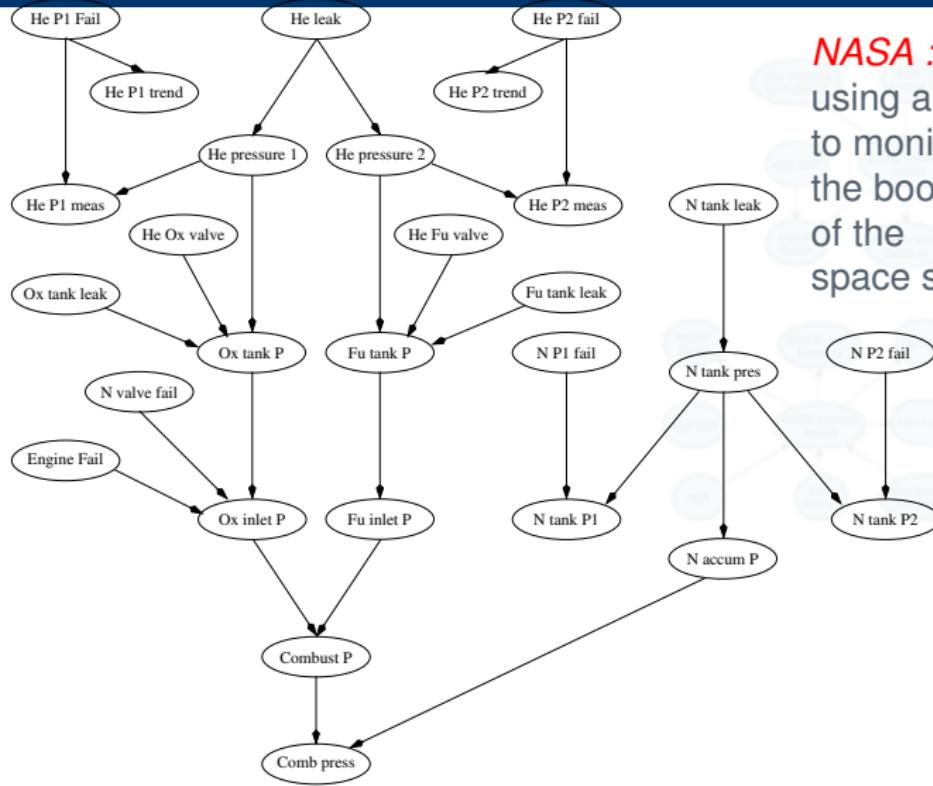
Application 1 : diagnostic

Diagnostic @ NASA



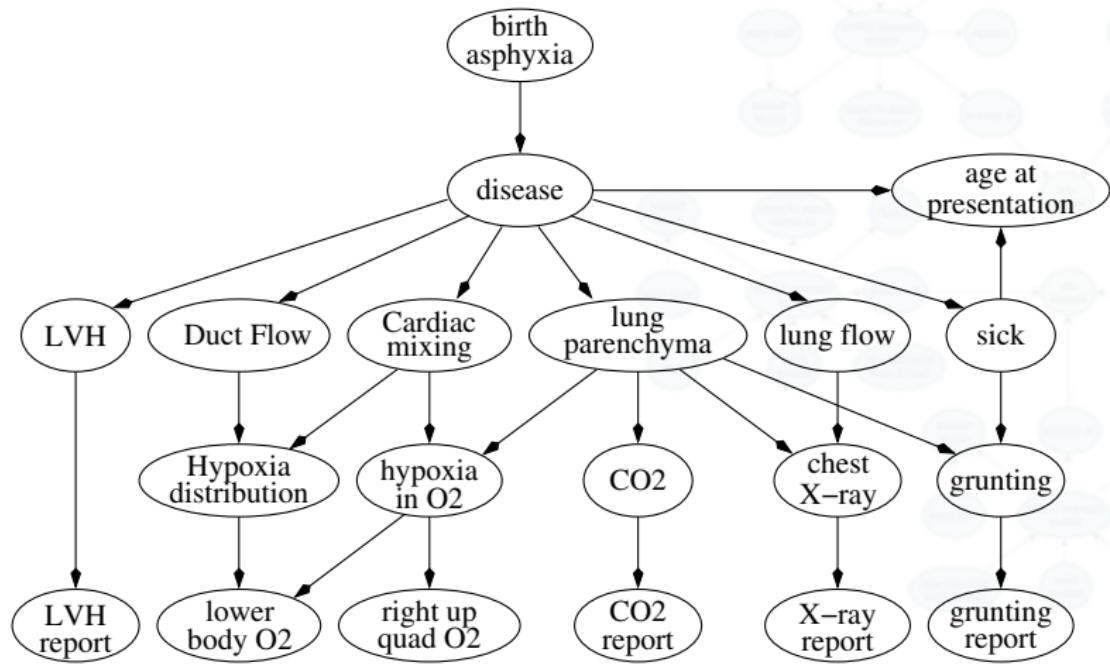
21

NASA :
using a BN
to monitor
the boosters
of the
space shuttle





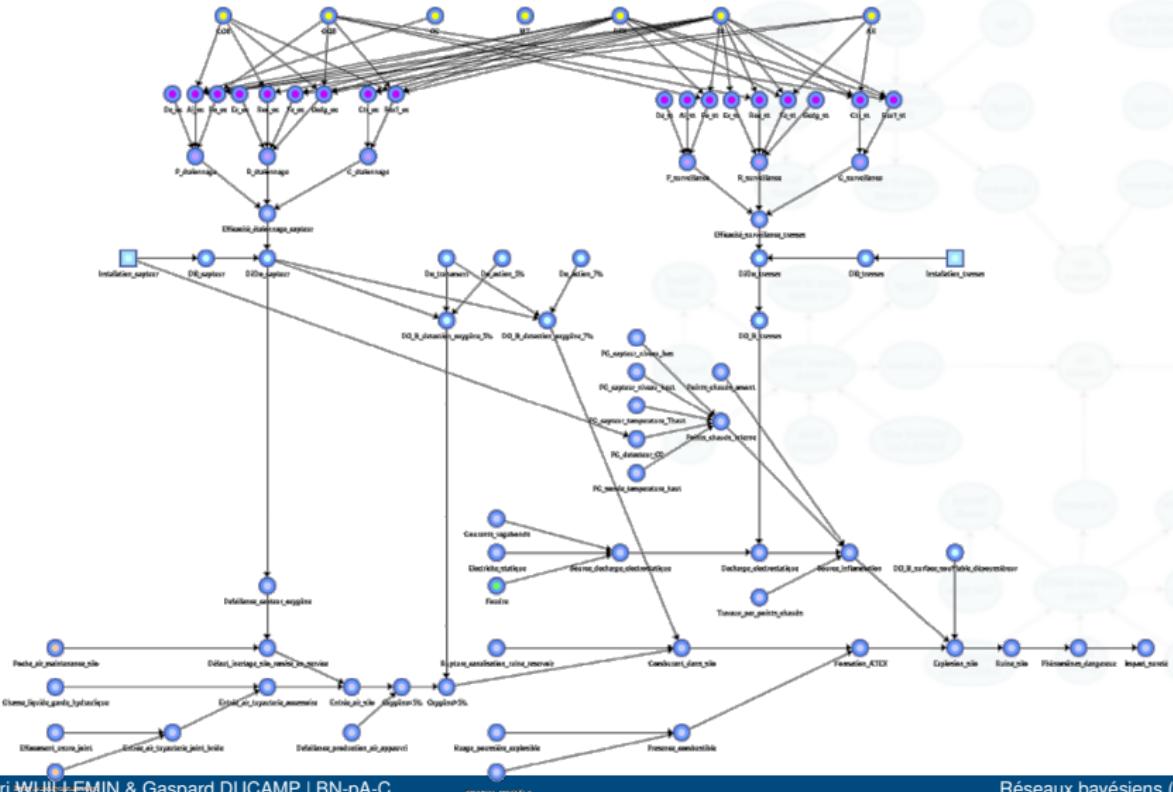
Diagnosis of the causes of cyanosis or heart attack in the child just after birth.



Application 3 : risk analysis



Risk modeling using BN : modular approach.



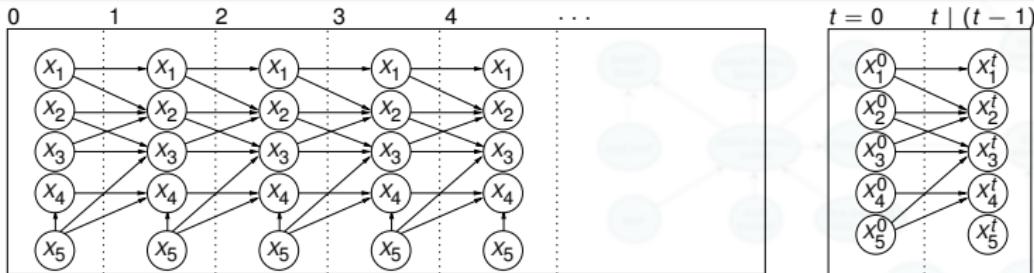
Application 4 : dynamic Bayesian networks



dBN (dynamic BN)

A dynamic BN is a BN where variables are indexed by the time t and by $i : X^t = \{X_1^t, \dots, X_N^t\}$ and verifies :

- ▶ Markov order 1 : $P(X^t | X^0, \dots, X^{t-1}) = P(X^t | X^{t-1})$,
- ▶ Homogeneity : $P(X^t | X^{t-1}) = \dots = P(X^1 | X^0)$.

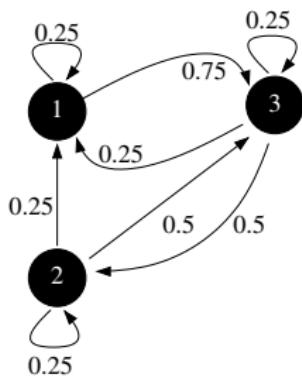


2-TBN

A dBN is characterized by :

- ▶ initial conditions ($P(X^0)$)
- ▶ the relations between $t - 1$ and t (*time-slice*).

2TBN (2 time-slices BN) allows to specify a dBN of size $T : X^0$ followed by T times ($t|t-1$).

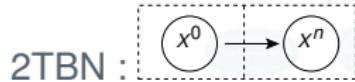
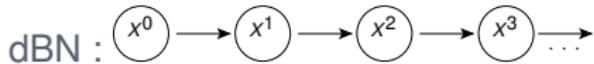


Markov chain

- ▶ A state variable (X^n) (at time n).
- ▶ Parameters :
 - ▶ Initial condition : $P(X^0)$
 - ▶ Transition model : $P(X^n|X^{n-1})$

$$P(X^n|X^{n-1}) = \begin{pmatrix} 0.25 & 0 & 0.75 \\ 0.25 & 0.25 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{pmatrix}$$

Equivalent dynamic Bayesian Network :

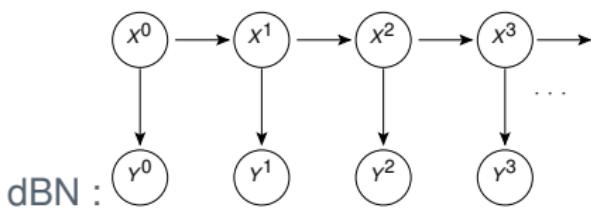




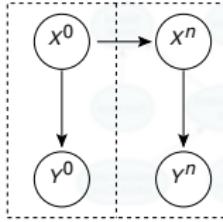
HMM

- ▶ A state variable (X^n) (at time n).
- ▶ An observation variable (Y^n)
- ▶ Parameters :
 - ▶ Initial condition : $P(X^0)$
 - ▶ Transition model : $P(X^n|X^{n-1})$
 - ▶ Observation model : $P(Y^n|X^n)$

Equivalent dynamic Bayesian Network :



2TBN :





Introduction

Réseaux bayésiens (discrets)

Definition

Inference

Applications

Use Cases

Dynamic Bayesian Networks

aGrUM/pyAgrum

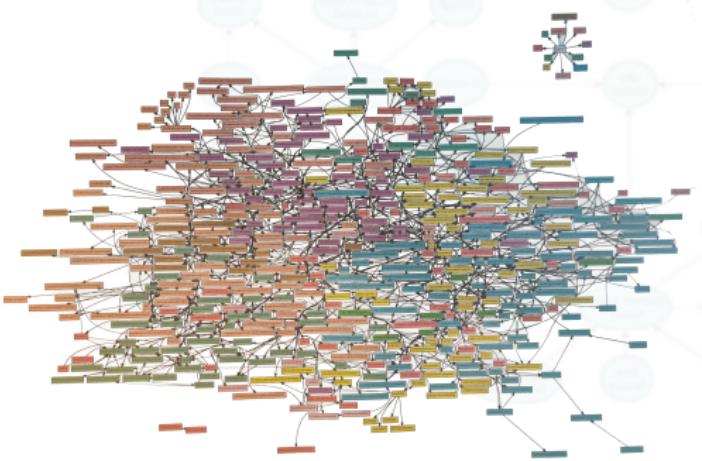
Learning Bayesian Networks





Un peu d'histoire

- ▶ The idea ? Build a set of common C++ codes for research team on PGM
- ▶ Became public and Open Source on GitLab in 2016
- ▶ 4800+ commits later thanks to 19 direct contributors...





aGrUM

Provides mainly low-level routines and components for PGM's algorithms but also high level components.

- ▶ Optimized implementation of core data structures
- ▶ Efficient implementation of state of the art algorithms
- ▶ High level components
- ▶ Built-in tools to ensure memory leak free code

☞ sophisticated but with a difficult learning curve

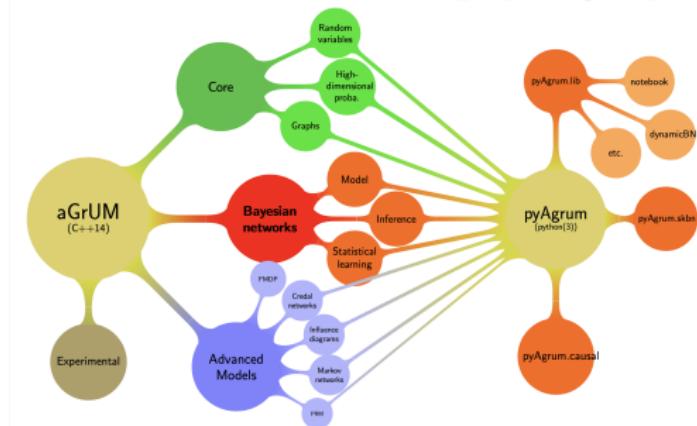
```
310 // diff the column types
311 const int64_t database = score_database..._databaseTable();
312 const std::size_t nb_vars = database.numberOfVariables();
313 const std::vector< gum::learning::DBTranslatedValueType > col_types;
314 nb_vars, value: gum::learning::DBTranslatedValueType::DISCRETE);
315
316 // create the bootstrap estimator
317 DBRowGenerator<CompleteRow> generator_bootstrap(col_types);
318 generator_bootstrap.set(gmset_bootstrap);
319 gmset_bootstrap.insertGenerator(generator_bootstrap);
320 DBRowGeneratorParser<*> parser_bootstrap(database.handler(),
321                                     gmset_bootstrap);
322 std::unique_ptr<ParamEstimator> parser_estimator_bootstrap(
323     createParamEstimator_( & parser_bootstrap, take_into_account_score));
324
325 // create the EM estimator
326 BayesNet< GUM_SCALAR > dummy_bn;
327 DBRowGenerator<*, GUM_SCALAR > generator_EM(col_types, dummy_bn);
328 DBRowGenerator<*> gen_EM = generator_EM; // fix for g++-4.8
329 DBRowGeneratorSet<*> gmset_EM;
330 gmset_EM.insertGenerator(generator_EM);
331 DBRowGeneratorParser<*> parser_database_handler(database.handler(), gmset_EM);
332 std::unique_ptr<ParamEstimator> parser_estimator_EM(
333     createParamEstimator_( & parser_EM, take_into_account_score));
```



pyAgrum

Simplified high-level API in an easier language (Python)

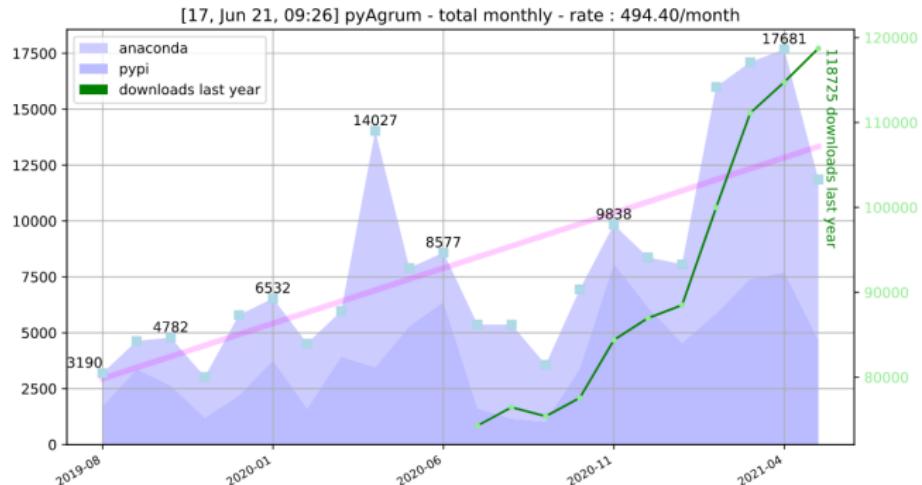
- ▶ Visualization tools (matplotlib + notebooks)
- ▶ Causal reasoning
- ▶ Scikit-Learn compliant classifiers





To sum up

- ☞ Using SWIG to wrap aGrUM
- ☞ Access to low-level methods and high-level tools
- ☞ Cross platform and cross Python
- ☞ Hundreds of unit tests





Model	Domain	Features
	▶ Input/Output	<input checked="" type="checkbox"/> bif/bifxml/dsl/net/uai/ o3prm formats
	▶ Exact Inference	<input checked="" type="checkbox"/> Variable Elimination, Shafer-Shenoy Inference, Lazy Propagation <input checked="" type="checkbox"/> Marginal targets, joint targets <input checked="" type="checkbox"/> Optimized Relevance Reasoning <input checked="" type="checkbox"/> Incremental inference
	▶ Approximated Inference	<input checked="" type="checkbox"/> Gibbs Sampling, Weighted Sampling, Importance Sampling <input checked="" type="checkbox"/> Loopy Belief Propagation <input checked="" type="checkbox"/> Gibbs, Weighted, Importance LoopySampling
● Bayesian Network	▶ Parameter Learning	<input checked="" type="checkbox"/> Pure max-Likelihood, Laplace, Dirichlet <input checked="" type="checkbox"/> Multiple score <input checked="" type="checkbox"/> Parametric EM for missing values.
	▶ Structural Learning	<input checked="" type="checkbox"/> score-based learning : Greedy Hill-Climbing, local search with tabu-list, K2 <input checked="" type="checkbox"/> information-based learning : 3off2, mlic (with latent confounder variable discovery) <input checked="" type="checkbox"/> Graphical constraints (forced arcs, forbidden arcs, initial structures, partial order, possible arcs)
	▶ Algorithms	<input checked="" type="checkbox"/> Exact and approximated distance/divergence between BNs (KL, Bhattacharya, Hellinger) <input checked="" type="checkbox"/> Mutual information, entropy <input checked="" type="checkbox"/> Simulation (generation of csv files) <input checked="" type="checkbox"/> Markov Blanket, essential graph etc.
● Markov network	▶ Input/Output	<input checked="" type="checkbox"/> uai
	▶ Inference	<input checked="" type="checkbox"/> Shafer-Shenoy
● Influence Diagram	▶ Input/Output	<input checked="" type="checkbox"/> bifxml
	▶ Inference	<input checked="" type="checkbox"/> Junction Trees
● Probabilistic Relational Model	▶ Input/output	<input checked="" type="checkbox"/> O3PRM language parser
	▶ Exact inference	<input checked="" type="checkbox"/> Structured Variable Elimination (SVE)
● Credal Networks	▶ Approximated inference	<input checked="" type="checkbox"/> GL2U, MC Sampling
	▶ Input	
● FMDP	▶ Planning	<input checked="" type="checkbox"/> SVI, SPUDD
	▶ Multi-Valued Decision Diagram	<input checked="" type="checkbox"/> SPUnDD



Applications

Used in many applications with partners :

- ▶ during PhDs and internships : IBM / Teranga / Airbus / ANR ...
- ▶ in collaboration in industrial projects
- ▶ by students in many courses





Pypi

Target : Python 2.8 / 3.6-9

Platform : win32, win64, Linux, OSX

```
pip install pyagrum
```

Conda

Target : Python 3.6-9

Platform : win64, Linux, OSX

```
conda install -c conda-forge pyagrum
```

+ Docker images, Binder ...



Manipulation de probabilités et pyAgrum :

- ▶ 00-tutorial
- ▶ 01-Probabilités
- ▶ 02-CPTdeterministe

Modélisation :

- ▶ 03-Modélisation1
- ▶ 04-Modélisation2
- ▶ 05-Modélisation3

Quelques ressources pour s'aiguiller :

- ▶ La doc de pyAgrum : <https://pyagrum.readthedocs.io/en/0.20.3/>
- ▶ Le notebook "cheatsheet" pour les manipulations de base



Introduction

Réseaux bayésiens (discrets)

Definition

Inference

Applications

Use Cases

Dynamic Bayesian Networks

aGrUM/pyAgrum

Learning Bayesian Networks





Apprentissage dans les réseaux bayésiens

L'apprentissage a pour but d'**estimer**, à partir d'une **base de données** et de **connaissances a priori** :

- ▶ La structure du réseau bayésien (X parent de Y ?)
- ▶ Les paramètres du réseau bayésien ($P(X = 0 \mid Y = 1)$?)

La base de données peut être :

- ▶ **complète**,
- ▶ **incomplète**.

Les connaissances a priori sont très variables ; par exemple :

- ▶ **structure du BN connue**,
- ▶ **Loi a priori pour certaines variables**, etc.

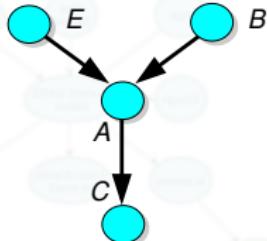
Ce qui donne 4 cadres principaux de l'apprentissage dans les réseaux Bayésiens :

“Apprentissage de {**paramètres** |structure} avec données {complètes |incomplètes}”.

Apprentissage des paramètres, données complètes



$$D : \begin{bmatrix} d_1^A & d_1^B & d_1^C & d_1^E \\ \dots & \dots & \dots & \dots \\ V & F & F & V \\ \dots & \dots & \dots & \dots \\ d_M^A & d_M^B & d_M^C & d_M^E \end{bmatrix}$$



En appelant Θ l'ensemble des paramètres du modèle et $L(\Theta : D)$ la vraisemblance :

$$L(\Theta : D) = P(D | \Theta)$$

$$= \prod_{m=1}^M P(d_m | \Theta)$$

(échantillons indépendants, identiquement distribués)

$$= \prod_{m=1}^M P(E = d_m^E, B = d_m^B, A = d_m^A, C = d_m^C | \Theta)$$

Apprentissage des paramètres, données complètes (2)

En renommant E, B, A, C par $n = 4, (X_i)_{1 \leq i \leq n}$,

$$\begin{aligned}L(\Theta : D) &= \prod_{m=1}^M P(X_1 = d_m^1, X_2 = d_m^2, \dots, X_n = d_m^n | \Theta) \\&= \prod_{m=1}^M \prod_{i=1}^n P(X_i | Pa_i, \Theta) \\&= \prod_{i=1}^n \prod_{m=1}^M P(X_i | Pa_i, \Theta_i) \\L(\Theta : D) &= \prod_{i=1}^n L_i(\Theta_i : D)\end{aligned}$$

L'estimation des paramètres d'un réseau bayésien se décomposent en l'estimation des paramètres de chaque loi de probabilité conditionnelle

Maximum de vraisemblance dans un réseau bayésien



$$\theta_{ijk} = P(X_i = k \mid Pa_i = j), N_{ijk} = \#_D(X_i = k, Pa_i = j), k \in \{1 \dots r_i\}, j \in \{1 \dots q_i\}$$

- $L(\Theta : D) = \prod_{i=1}^n L_i(\Theta_i : D) = \prod_{i=1}^n \prod_{m=1}^M P(X_i = k_m \mid Pa_i = j_m, \Theta_i)$

$$L(\Theta : D) = \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}}$$

- $LL(\Theta : D) = \sum_{i=1}^n \sum_{m=1}^M \log P(X_i \mid Pa_i, \Theta_i) = \sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \theta_{ijk}$

- On sait que $\sum_k \theta_{ijk} = 1$ soit $\theta_{ijr_i} = 1 - \sum_{k=1}^{r_i-1} \theta_{ijk}$ d'où

$$LL(\Theta : D) = \sum_{i=1}^n \sum_{j=1}^{q_i} \left(\sum_k^{r_i-1} N_{ijk} \log \theta_{ijk} + N_{ijr_i} \log \left(1 - \sum_{k=1}^{r_i-1} \theta_{ijk} \right) \right)$$

- On cherche $\hat{\Theta}$ maximisant $L(\Theta : D)$ et donc $LL(\Theta : D)$:

i.e. $\hat{\Theta}$ tel que $\forall i, \forall j, \forall k$,

$$\frac{\partial LL(\Theta : D)}{\partial \theta_{ijk}} (\hat{\Theta}) = \frac{N_{ijk}}{\hat{\theta}_{ijk}} - \frac{N_{ijr_i}}{1 - \sum_{k=1}^{r_i-1} \hat{\theta}_{ijk}} = \frac{N_{ijk}}{\hat{\theta}_{ijk}} - \frac{N_{ijr_i}}{\hat{\theta}_{ijr_i}} = 0$$

- Finalement, $\frac{N_{ijr_i}}{\hat{\theta}_{ijr_i}} = \frac{N_{ij1}}{\hat{\theta}_{ij1}} = \dots = \frac{N_{ij(r_i-1)}}{\hat{\theta}_{ij(r_i-1)}}$ (et $\sum_k \hat{\theta}_{ijk} = 1$) d'où

$$\forall k \in \{1, \dots, r_i\}, \hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$$

avec $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$

Prédiction bayésienne



Θ suit une distribution $P(\Theta \mid D)$.

$$P(\Theta \mid D) \propto P(D \mid \Theta) \cdot P(\Theta) = L(\Theta : D) \cdot P(\Theta)$$

Cette méthode permet de prendre en compte un *a priori* sur Θ ; pour intégrer des connaissances d'expert ou pour rendre plus stable les estimations avec un petit échantillon D .

Distribution de Dirichlet :

$$f(p_1, \dots, p_K; \alpha_1, \dots, \alpha_K) \propto \prod_{i=1}^K x_i^{\alpha_i - 1}$$

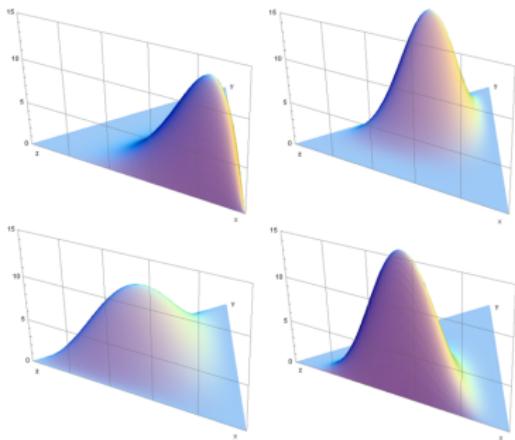
(où $\sum_i p_i = 1$)

Intuitivement, f se lit comme :

$$P(P(X = i) = p_i \mid \#x=i = \alpha_i - 1)$$

En supposant que l'a priori $P(\Theta)$ soit une distribution de Dirichlet :

$$P(\Theta) = \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_j} \theta_{ijk}^{\alpha_{ijk} - 1}$$



[Wikipedia] Clockwise from top left :

$$\alpha = (6, 2, 2), (3, 7, 5), (6, 2, 6), (2, 3, 4).$$

Prédiction bayésienne (2)



40

À partir de :

$$\bullet P(\Theta \mid D) \propto L(\Theta : D) \cdot P(\Theta)$$

$$\bullet P(\Theta) = \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_j} \theta_{ijk}^{\alpha_{ijk}-1}$$

$$\bullet L(\Theta : D) = \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_j} \theta_{ijk}^{N_{ijk}}$$

$$P(\Theta \mid D) = \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_j} \theta_{ijk}^{N_{ijk} + \alpha_{ijk} - 1}$$

MAP : maximum a posteriori

$$\hat{\Theta}^{\text{MAP}} = \arg \max_{\Theta} P(\Theta \mid D)$$

$$\hat{\theta}_{ijk}^{\text{MAP}} = \frac{N_{ijk} + \alpha_{ijk} - 1}{\sum_k (N_{ijk} + \alpha_{ijk} - 1)}$$

EAP : espérance a posteriori

$$\hat{\Theta}^{\text{EAP}} = \int_{\Theta} \Theta \cdot P(\Theta \mid D) d\Theta$$

$$\hat{\theta}_{ijk}^{\text{EAP}} = \frac{N_{ijk} + \alpha_{ijk}}{\sum_k (N_{ijk} + \alpha_{ijk})}$$

Apprentissage des paramètres, données complètes

Résumé



Avec N_{ijk} le nombre de fois où la variable X_i a pris la valeur k et ses parents la valeur (t-uple) j et α_{ijk} les paramètres d'un a priori de Dirichlet.

Estimation des paramètres

Deux méthodes possibles pour l'estimation des paramètres :

- MLE (Maximum Likelihood Estimation)

$$\hat{\theta}_{ijk} = \hat{\theta}_{\{x_i=k|pa_i=j\}} = \frac{N_{ijk}}{N_{ij}}$$

- Estimation bayésienne (avec *a priori* de Dirichlet)

$$\hat{\theta}_{ijk}^{MAP} = \hat{\theta}_{\{x_i=k|pa_i=j\}} = \frac{\alpha_{ijk} + N_{ijk} - 1}{\alpha_{ij} + N_{ij} - r_i}$$

$$\hat{\theta}_{ijk}^{EAP} = \hat{\theta}_{\{x_i=k|pa_i=j\}} = \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}$$

- *A priori* important quand $N_{ijk} \rightarrow 0$: pas de cas dans la base.
- Les estimations sont consistantes et équivalentes quand $N_{ijk} \rightarrow \infty$



Des correctifs 'pragmatiques' ont été proposés dans le cas où peu de données rendaient l'estimation des paramètres fragiles.

Ajustement des paramètres (éviter les 0)

- **a priori de Dirichlet** $\widehat{\theta}_{ijk} \approx \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}}$ avec $\alpha_{ij} = \sum_k \alpha_{ijk}$

PS- α_{ij} est à comparer à N_{ij} : elle détermine l'influence a l'*a priori* sur la loi.

- **ajustement de Laplace** $\widehat{\theta}_{ijk} \approx \frac{N_{ijk} + 1}{N_{ij} + |X_i|}$

PS- revient au cas précédent avec $\alpha_{ijk} = 1$: a priori uniforme, influence faible.

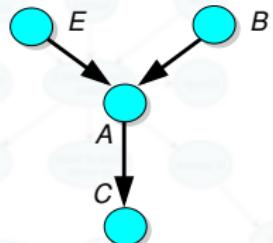
- **actualisation de Ney-Essen**

On retire à tout x une valeur fixe δ et on répartit uniformément la somme collectées.

$$D_{ij} = \sum_k \min(N_{ijk}, \delta) \quad \text{et} \quad \widehat{\theta}_{ijk} \approx \frac{N_{ijk} - \min(N_{ijk}, \delta) + \frac{D_{ij}}{|X_i|}}{N_{ij}}$$



$$D : \begin{bmatrix} d_1^A & d_1^B & d_1^C & d_1^E \\ \dots & \dots & \dots & \dots \\ V & F & ? & V \\ V & F & ? & V \\ ? & F & ? & V \\ \dots & \dots & \dots & \dots \\ d_M^A & d_M^B & d_M^C & d_M^E \end{bmatrix}$$



$D = D^o \cup D^h$ respectivement données observées et données manquantes.

Typologie des données incomplètes

En notant $\mathcal{M}_{il} = (d_{il}^j \in D^h)$

- ▶ MCAR : $P(\mathcal{M} | D) = P(\mathcal{M})$ (Missing Completely At Random).
- ▶ MAR : $P(\mathcal{M} | D) = P(\mathcal{M} | D^o)$ (Missing At Random).
- ▶ NMAR : $P(\mathcal{M} | D)$ (Not Missing At Random).

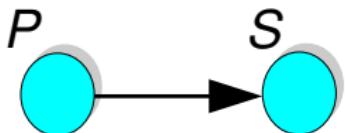


EM dans les BNs

Répéter jusqu'à convergence

Étape E : Estimer $N_{ijk}^{(t+1)}$ à partir des $P(X_i | Pa_i, \theta_{ijk}^t)$
inférence dans le BN de paramètres θ_{ijk}^t

Étape M : $\theta_{ijk}^{t+1} = \frac{N_{ijk}^{(t+1)}}{N_{ij}^{(t+1)}}$



P	S
o	?
n	?
o	n
n	n
o	o

Paramètres à estimer :

- $P(P) = [\theta_P \ 1 - \theta_P]$
- $P(S | P = o) = [\theta_{S|P=o} \ 1 - \theta_{S|P=o}]$
- $P(S | P = n) = [\theta_{S|P=n} \ 1 - \theta_{S|P=n}]$

Par MLE : $\theta_P = \frac{3}{5}$



- ▶ **But :** obtenir automatiquement une structure de réseau bayésien à partir de données.
- ▶ **En théorie :** Test du χ^2 plus énumération de tous les modèles possibles : OK
- ▶ **En pratique :** Beaucoup de problème mais avant tout :

Espace des réseaux bayésiens (Robinson, 1977)

Le nombre de structures possibles pour n nœuds est super-exponentiel.

$$NS(n) = \begin{cases} 1 & , n \leq 1 \\ \sum_{i=1}^n (-1)^{i+1} \cdot C_i^n \cdot 2^{i \cdot (n-i)} \cdot NS(n-1) & , n > 1 \end{cases}$$

Robinson (1977) *Counting unlabelled acyclic digraphs*. In Lecture Notes in Mathematics : Combinatorial Mathematics V

La recherche exhaustive n'est pas possible. L'espace est bien trop grand : $NS(10) \approx 4.2 \cdot 10^{18}$!



Tableau général de l'apprentissage

Recherche de relation symétrique + orientation (*causalité*)

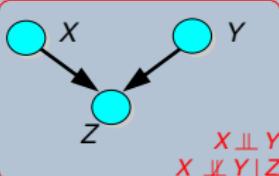
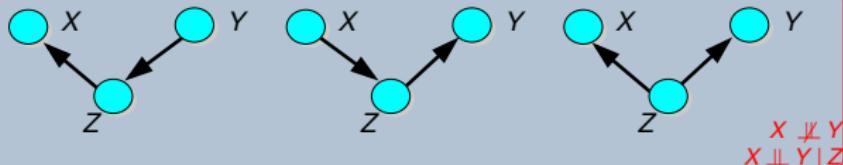
- ▶ algorithme IC/PC
- ▶ algorithme IC*/FCI

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, ...).

Classe d'équivalence de Markov

Deux réseaux bayésiens sont équivalents si ils représentent le même modèle d'indépendance.

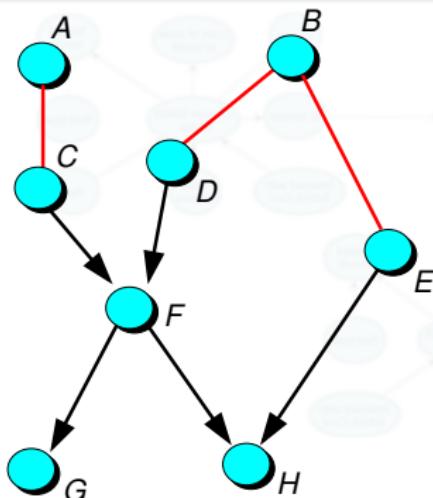
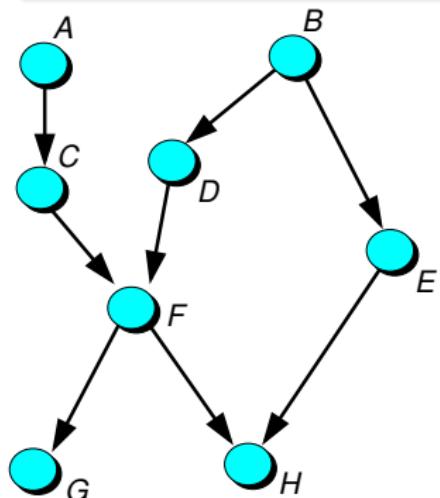




Classe d'équivalence de Markov, graphe essentiel

Une **classe d'équivalence de Markov** est l'ensemble de réseaux bayésiens qui sont tous équivalents.

Elle peut être représentée par le graphe sans circuit partiellement orienté qui a la même structure que tous les réseaux équivalents, mais pour lequel les arcs réversibles (n'appartenant pas à des V-structures, ou dont l'inversion ne génère pas de V-structure) sont remplacés par des arêtes (non orientées) : le **graphe essentiel**.





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. Déetecter les V-structures et les orientations qu'elles impliquent.
3. Finaliser les orientations en restant dans la même classe d'équivalence de Markov.

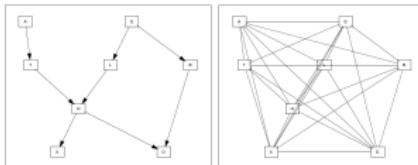
Écueils principaux : un très grand nombre de tests d'indépendances, chaque test étant très sensible au nombre de données disponibles.

Exemple PC



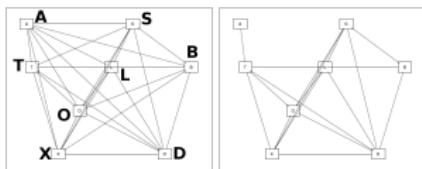
- Soit un réseau bayésien (à gauche) qui a permis de créer une base de 5000 cas.¹

Etape 0 : Graphe non orienté reliant tous les nœuds.



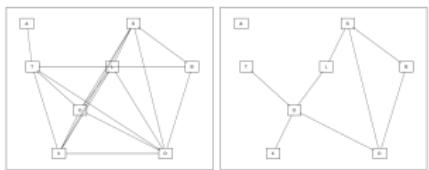
- Par des χ^2 , on teste toutes les indépendances marginales ($X \perp\!\!\!\perp Y$) puis les indépendances par rapport à une variable ($X \perp\!\!\!\perp Y | Z$).

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



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

Etape 1b : Suppression des ind. conditionnelles d'ordre 1



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

1. Exemple de Philippe Leray

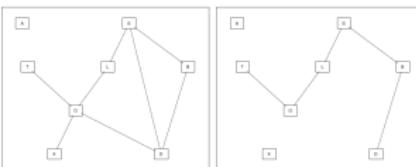
Exemple PC



50

- On continue les χ^2 d'ordre supérieur

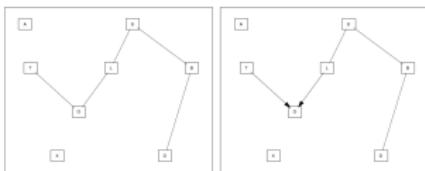
Etape 1c : Suppression des ind. conditionnelles d'ordre 2



On trouve : $D \perp\!\!\!\perp S | (L, B)$, $X \perp\!\!\!\perp O | (T, L)$, $D \perp\!\!\!\perp O | (T, L)$.

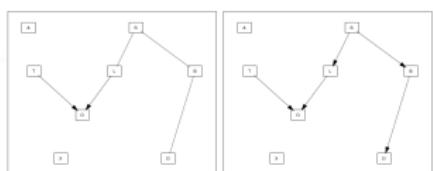
- Recherche des V-Structure, propagation des contraintes d'orientations puis orientations des dernières arêtes en restant Markov-équivalent.

Etape 2 : Recherche des V-structures



On trouve : $T \not\perp\!\!\!\perp L$ et $T \perp\!\!\!\perp L | O$

Etape 4 : Instantiation du PDAG



Orientation sans nouvelle V-structure

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

Recherche locale à base de scores



La recherche exhaustive des relations d'indépendances est inatteignable (nombre de tests prohibitifs, quantité de données nécessaires trop importantes, etc.). Donc utilisation d'une heuristique permettant de quantifier l'adéquation d'une structure à une base de données. L'algorithme de recherche locale est un algorithme générique qui ne demande que quelques hypothèses de base :

Recherche locale

- ▶ Soit un espace de recherche,
- ▶ Soit une notion de voisinage définie par des opérations élémentaires (les voisins d'un élément sont les points atteignables par l'application d'une opération élémentaire à cet élément).
- ▶ Soit un score (heuristique) calculable localement.
- ▶ La recherche locale est alors une séquence de voisins tels qu'à partir du point initial, tout élément ultérieur de la séquence augmente le score. (*Greedy Search*).

Recherche locale dans les réseaux bayésiens

- ▶ L'espace est l'espace des réseaux bayésiens (énorme)
- ▶ Le score : voir slides suivants
- ▶ Soit une structure initiale
- ▶ Les opérations de base : ajout/suppression/modification d'un arc (dans le domaine de validité)



Propriétés des scores

Soient D la base de donnée, T la topologie du réseau bayésien candidat et Θ ses paramètres. Pour qu'un score (une fonction calculée sur un réseau bayésien) soit considéré comme une bonne heuristique, on peut lui demander :

1. **Vraisemblance** : Coller le mieux aux données ($\max L(T, \Theta : D)$).
2. **Rasoir d'Occam** : Privilégier les topologies T simples aux topologies complexes ($\min \text{Dim}(T)$).
3. **Consistance locale** : Ajouter un arc 'utile' devrait augmenter le score. Ajouter un arc 'inutile' devrait diminuer le score.
4. **Score équivalence** : Deux réseaux bayésiens Markov-équivalents devraient avoir le même score.
5. **Décomposition locale** : Calculer la modification du score par l'ajout/retrait d'un arc ne doit pas imposer de re-calculer tout le score mais seulement une partie, locale à l'arc modifié.

Quelques scores (1) : AIC/BIC



Idée de base : Il faut maximiser la vraisemblance tout en minimisant la dimension.

Score AIC (Akaike, 70)

- Akaike Information Criterion

$$\text{Score}_{\text{AIC}}(T, D) = \log_2 L(\Theta^{\text{MV}}, T : D) - \text{Dim}(T)$$

Score BIC (Schwartz, 78)

- Bayesian Information Criterion

$$\text{Score}_{\text{BIC}}(T, D) = \log_2 L(\Theta^{\text{MV}}, T : D) - \frac{1}{2} \cdot \text{Dim}(T) \cdot \log_2 N$$



MDL consiste à considérer la compacité de la représentation du modèle comme un bon critère de la qualité de ce modèle. Étonnamment, ce critère est équivalent au critère BIC ci-dessus. Il s'agit donc de minimiser la taille de la représentation, composée de :

- ▶ la représentation du modèle,
- ▶ la représentation des données sous forme de paramètres du modèle.

Score MDL (Lam and Bacchus, 93)

● Minimum Description Length

$$\text{Score}_{\text{MDL}}(T, D) = \log_2 L(\Theta^{\text{MV}}, T : D) - |\text{arcs}_T| \cdot \log_2 N - c \cdot \text{Dim}(T)$$

où arcs_T est l'ensemble des arcs du graphe, c est le nombre de bits nécessaire à la représentation d'un paramètre.

Quelques scores (3) : BDe



Avec un critère bayésien, il s'agirait simplement de maximiser la probabilité jointe de T et D :

$$\begin{aligned} P(T, D) &= \int_{\Theta} P(D | \Theta, T) \cdot P(\Theta | T) \cdot P(T) d\Theta \\ &= P(T) \cdot \int_{\Theta} L(\Theta, T : D) \cdot P(\Theta | T) d\Theta \end{aligned}$$

Avec des hypothèses d'indépendances, un *a priori* de Dirichlet bien choisi, on obtient :

Score BDe

- Bayesian Dirichlet score Equivalent

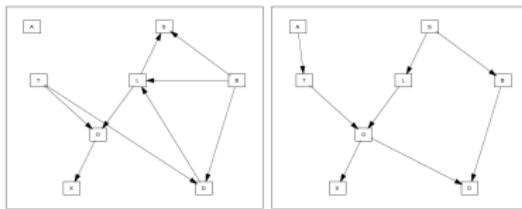
$$\text{Score}_{\text{BDe}}(T, D) = P(T) \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{i,j})}{\Gamma(N_{i,j} + \alpha_{i,j})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{i,j,k} + \alpha_{i,j,k})}{\Gamma(\alpha_{i,j,k})}$$

Recherche locale : Greedy Search



Algorithme implémentant exactement ce qui est défini précédemment.

Réseau obtenu vs. théorique



L'algorithme peut être bloqué sur des 'plateaux' et/ou converger vers des minima locaux.

Solutions

Principalement des méthodes de méta-heurisitiques :

- ▶ Random restart
- ▶ TABU-search (liste des K dernières structures à éviter)
- ▶ Simulated annealing (accepter des structures diminuant le score avec un seuil diminuant au cours du temps)

Recherche locale : Diminution de l'espace de recherche



57

S'il existe un ordre dans les nœuds, tel qu'il ne soit pas possible d'avoir des arcs rétrogrades, alors il y a diminution de la taille de l'espace de recherche.

Taille de l'espace de recherche avec ordre sur les nœuds

$$NS'(n) = 2^{\frac{n \cdot (n-1)}{2}}$$

Algorithme K2

- ▶ Réseau initial sans arcs
- ▶ Opération élémentaire : ajouter un arc de j à $i > j$.
- ▶ Greedy algorithm sur le score BD(e).
- ▶ Limite sur le nombre de parents maximum.

Problème principal : algorithme très dépendant de l'ordre.

Réduction de l'espace de recherche : Équivalents de Markov

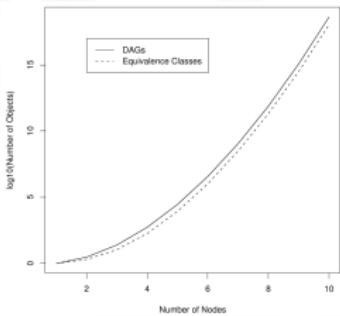


Comme la recherche locale peut tomber dans des minima locaux, l'idée est de s'affranchir d'une partie de ces minima en changeant d'espace pour l'espace des classes d'équivalence de Markov.

Greedy Equivalence Search

L'espace des classes d'équivalences (notés les graphes essentiels) a une structure. On peut définir des opérateurs élémentaires et donc mener une recherche locale.

- ▶ Avantage : Pas de plage de score équivalence.
- ▶ Pas avantage : La taille de l'espace de recherche est sensiblement la même (ratio asymptotique de 3.7).



Réduction de l'espace de recherche : Recherche dans les arbres



Cette recherche se limite aux BNs dans lesquels chaque nœud a au plus un parent.

Malgré une simplification (trop) grande du modèle, les arbres apportent :

- ▶ une solution élégante mathématiquement (optimisation globale),
- ▶ un nombre de paramètres minimum (minimise le risque de sur-apprentissage).

La décomposabilité du score donne :

$$LL(T) = \sum_i LL_i(i, pa(i)) = \sum_{X \rightarrow Y} LL(Y \leftarrow X) + K$$

$$\text{avec } LL(Y \leftarrow X) = LL_Y(Y, X) - LL_Y(Y, \emptyset)$$

Recherche de l'arbre de **score maximal**

- ▶ $\forall X, Y$, calculer $LL(Y \leftarrow X)$
- ▶ Trouver l'arbre (ou la forêt) de poids maximal.
Max Spanning Tree Algorithm – $O(n^2 \cdot \log(n))$

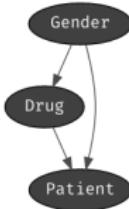


Retour sur le paradoxe de Simpson



60

```
learner=gum.BNLearner("simpson.csv")
bn=learner.learnBN()
gnb.sideBySide(bn,[bn cpt(x) for x in bn.nodes()])
```



		Drug	
Gender	With	Without	
F	0.7675	0.2325	
M	0.2468	0.7532	

Gender		
	F	M
	0.5080	0.4920

		Patient	
Drug	Gender	Healed	Sick
With	F	0.6683	0.3317
	M	0.1982	0.8018
Without	F	0.7793	0.2207
	M	0.3993	0.6007

```
ie=gum.LazyPropagation(bn)
gnb.sideBySide(ie.evidenceImpact(target="Patient",evs="Drug"),ie.evidenceImpact(target="Patient",evs=["Drug","Gender"]))
```

		Patient	
Drug	Healed	Sick	
With	0.5567	0.4433	
Without	0.4911	0.5089	

		Patient	
Drug	Gender	Healed	Sick
With	F	0.6683	0.3317
	M	0.1982	0.8018
Without	F	0.7793	0.2207
	M	0.3993	0.6007

Conclusions sur Simpson



Quoi ?



Comment ?



Et donc ?

What do we gain with graphical Models ?



- ▶ A *compact model* : gains in time and space (tractable **exact inference, approximated inference, sampling**), etc.
- ▶ A *learnable model*
- ▶ A *qualitative knowledge discovery* from data,
 - ▶ Validation,
 - ▶ Prediction,
 - ▶ Explicability, etc.
- ▶ And even, a possible *causal approach* from data.
- ▶ However,  still a NP-hard problem !



Modélisation et apprentissage :

- ▶ 06-ModelSelection
- ▶ 07-Learning

Pour les curieux et curieuses qui souhaiteraient en voir plus :

- ▶ 08-KaggleTitanic

Quelques ressources pour s'aiguiller :

- ▶ La doc de pyAgrum : <https://pyagrum.readthedocs.io/en/0.20.3/>
- ▶ Le notebook "cheatsheet" pour les manipulations de base