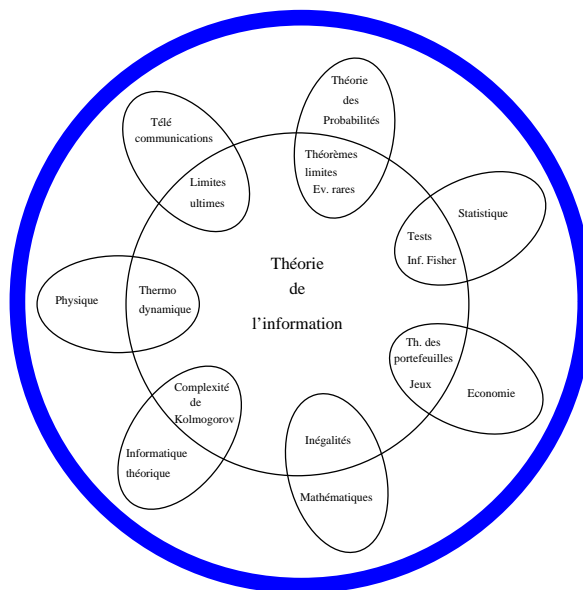


THÉORIE DE L'INFORMATION ET DU CODAGE



NOTES RELATIVES AUX TRAVAUX PRATIQUES

Note. La version pdf de ce document ne s'imprime pas toujours correctement. Une version ps est disponible au lien Page WEB du cours "Théorie de l'information et du codage"

Olivier BILENNE
Jérôme DELVAUX
Pierre GEURTS
Marc-Antoine LEMAIRE
Julien MICHEL
Patricia ROUSSEAU
Louis WEHENKEL

Novembre 2002

Table des matières

1. INTRODUCTION	1
1.1 Programme mis à disposition	1
1.2 Organisation	1
1.3 Evaluation	2
2. ENONCES DES TRAVAUX	3
2.1 Modélisation probabiliste de sources d'information	3
2.1.1 Modélisation	3
2.1.2 Analyse des propriétés de stationnarité et d'ergodicité	4
2.1.3 Calculs d'entropie	4
2.2 Codage/décodage de canal	5
2.2.1 Source sans mémoire, bruit sans mémoire	5
2.2.2 Source sans mémoire, bruit de canal avec mémoire	6
2.2.3 Symboles source corrélés, bruit sans mémoire	7
3. SIMULATION DE CHAÎNES DE MARKOV AVEC JAVABAYES	9
3.1 Sélection de variables	9
3.2 Création de variables identités	9
3.3 Calcul d'entropie conditionnelle	9
3.4 Simulation de variables	10
3.5 Création de chaînes de Markov de mémoire 1	10
3.6 Simulation de chaînes de Markov	10

1 INTRODUCTION

Ces notes reprennent l'ensemble des informations nécessaires aux travaux pratiques sur ordinateur du cours de théorie de l'information et du codage.

Pour l'année académique 2002-2003, le travail comporte deux parties : la première porte sur la modélisation de sources (processus de Markov), la seconde porte sur l'étude du codage de canaux. Les deux parties font appel au logiciel "JavaBayes".

Le but de ces travaux pratiques est de permettre aux étudiants de manipuler de façon concrète les notions théoriques introduites dans ce cours grâce à différentes expériences.

1.1 PROGRAMME MIS À DISPOSITION

Le programme "JavaBayes" de simulation et de manipulation de réseaux bayésiens a été modifié pour les besoins de ce travail (cfr chapitre 3). Il est accessible au moyen de la commande suivante

```
/home/lwh/JavaBayes
```

Ce logiciel permet de créer des réseaux Bayésiens et de résoudre différents problèmes d'inférence.

Les fichiers de description de réseaux bayésiens utilisés dans le cadre du travail pratique se trouvent dans le répertoire suivant

```
/home/lwh/JBex/
```

Le programme se manipule au moyen d'une interface graphique relativement intuitive. Des informations plus détaillées se trouvent sur la page WEB <http://www.montefiore.ulg.ac.be/~lwh/javabayes/> ainsi qu'au chapitre 3.

1.2 ORGANISATION

- Les travaux doivent être réalisés par groupe de deux étudiants.
- Les groupes doivent être constitués au plus tard pour le 29 novembre 2002.
- Une fois constitué, le groupe enverra un e-mail à l'adresse "P.Rousseaux@ulg.ac.be" signalant les noms des étudiants constituant ce groupe ainsi qu'une adresse e-mail. L'énoncé du travail sera envoyé à cette adresse.

On demande aux étudiants de remettre un rapport écrit décrivant, discutant et justifiant de façon suffisamment détaillée les résultats obtenus par expérimentation. Le but est de montrer qu'on a bien compris les notions du cours sur lesquelles porte le travail.

Les fichiers de description des réseaux bayésiens mis au point doivent accompagner ce document. L'ensemble de ces documents doivent être transmis par email à "P.Rousseaux@ulg.ac.be" et une copie papier du rapport doit être remise à la date limite.

Le date limite de remise des travaux est fixée au lundi 6 janvier 2003.

1.3 EVALUATION

Les travaux pratiques donneront lieu à une cote qui interviendra pour un tiers dans la cote globale du cours de théorie de l'information et du codage. Les étudiants seront interrogés sur le travail lors de l'examen.

2 ENONCES DES TRAVAUX

2.1 MODÉLISATION PROBABILISTE DE SOURCES D'INFORMATION

Cette partie du travail se focalise sur les notions de processus et de variables aléatoires discrètes et les notions d'entropie et d'information mutuelle introduites aux chapitres 4 et 8 du cours.

Il s'agit d'utiliser le logiciel "JavaBayes" afin de simuler différentes sources d'information et de vérifier leurs principales propriétés (génération de réalisations de longueur quelconque, étude des propriétés statistiques de ces réalisations, entropies par symbole, stationnarité, ergodicité).

On considère les sources binaires d'information suivantes :

- source S_1 : source sans mémoire, dont les symboles sont distribués selon $P(s_i = 0) = 0.6 \quad \forall i > 0$
- source S_2 : source de Markov de mémoire 1 dont les probabilités de transition sont définies $\forall i > 0$ par :

$$\begin{aligned}P(s_{i+1} = 1 | s_i = 1) &= 0.7 \\P(s_{i+1} = 0 | s_i = 0) &= 0.7\end{aligned}$$

- source S_3 : source de Markov de mémoire 2 dont les probabilités de transition sont définies $\forall i > 1$ par :

$$\begin{aligned}P(s_{i+1} = 1 | s_i = 1, s_{i-1} = 1) &= 0.8 \\P(s_{i+1} = 0 | s_i = 0, s_{i-1} = 0) &= 0.8 \\P(s_{i+1} = 0 | s_i = 1, s_{i-1} = 0) &= 0.5 \\P(s_{i+1} = 0 | s_i = 0, s_{i-1} = 1) &= 0.5\end{aligned}$$

2.1.1 Modélisation

Utiliser le programme "JavaBayes" pour représenter chacune de ces sources par un réseau bayésien. On se limitera à la représentation des 5 premiers symboles émis. Le choix des distributions de probabilités initiales est libre. Remarquons que l'initialisation de la source S_3 requiert de fixer les deux distributions $P(S_1)$ et $P(S_2|S_1)$.

A l'aide de ces réseaux, vérifier :

1. que la source S_1 émet bien des symboles successifs indépendants et identiquement distribués
2. que la source S_2 constitue bien une chaîne de Markov de mémoire 1; déterminer le diagramme et la matrice de transition d'états de cette chaîne ainsi que la (ou les) distributions stationnaires;
3. que la source S_3 constitue bien une chaîne de Markov de mémoire 2; déterminer le diagramme et la matrice de transition d'états de cette chaîne ainsi que la (ou les) distributions stationnaires.

Les états d'une chaîne de Markov de mémoire 2 peuvent être définis de deux manières :

- (a) à partir de l'extension d'ordre deux de la source. Dans ce cas, les états sont définis par les réalisations des doublets s_{2i}, s_{2i-1}
- (b) par les réalisations de toutes les séquences de deux symboles successifs s_{i+1}, s_i .

Considérer ces deux approches pour déterminer le diagramme et la matrice de transition d'états. Montrer (expérimentalement ou par calcul) que l'on obtient bien dans les deux cas les mêmes distributions stationnaires.

Le choix des distributions de probabilités initiales est libre. Remarquons que l'initialisation de la source S_3 requiert de fixer les deux distributions $P(S_1)$ et $P(S_2|S_1)$.

2.1.2 Analyse des propriétés de stationnarité et d'ergodicité

Pour tester ces propriétés, on exploitera les fonctionnalités de "JavaBayes" permettant de générer automatiquement des chaînes de Markov de mémoire 1 de longueur déterminée (cfr Chapitre 3). Il importera donc de transformer tout d'abord les sources définies ci-dessus en chaînes de Markov de mémoire 1.

1. Vérifier, en comparant les probabilités de patterns décalés dans le temps, sous quelles conditions les trois sources sont stationnaires.
2. Vérifier, en comparant les probabilités de patterns avec les statistiques temporelles, si, sous les conditions du point 1, les trois sources sont ergodiques.
3. Utiliser le programme pour trouver des régimes permanents. Comparer les régimes permanents obtenus (s'il y en a) pour différentes distributions de probabilités initiales, et observer différentes réalisations pour chaque cas. Comparer les régimes permanents avec les distributions stationnaires.
4. Transformer la chaîne de Markov de départ S_3 en une nouvelle chaîne S_4 réductible et qui possède plus d'une distribution stationnaire.
5. Déterminer le diagramme et la matrice de transition d'états de cette nouvelle source ainsi que ses distributions stationnaires.
6. Répéter les points 1 à 3 ci-dessus pour cette nouvelle source.

2.1.3 Calculs d'entropie

Calculer les entropies suivantes pour chacune des sources S_1 à S_4 (initialisées avec leur distribution stationnaire). Par symbole, on entend les symboles binaires 0, 1.

1. Entropie conjointe des k premiers symboles (tester avec JavaBayes pour $k = 2, 4, 5$).
2. Entropie conditionnelle moyenne du $k + 1$ -ème symbole connaissant les k premiers (tester pour $k = 1, 2, 4, 5$).
3. Information mutuelle entre le $k + 1$ -ème symbole et les k premiers (tester pour $k = 1, 2, 4, 5$).
4. Information mutuelle entre le premier symbole et les k suivants (tester pour $k = 1, 2, 4, 5$).
5. L'entropie de la source $H(S)$.

Discuter les résultats obtenus et justifier.

2.2 CODAGE/DÉCODAGE DE CANAL

Cette partie repose sur l'utilisation de réseaux bayésiens pour la modélisation d'une communication au travers d'un canal.

Une source S émet aléatoirement des séquences de symboles binaires $s_1, s_2, s_3, s_4 \dots$. Les bits source sont codés et transmis sur un canal binaire selon le schéma suivant :

- pour chaque bit source s_i deux symboles binaires $x_{i,1}$ et $x_{i,2}$ seront produits au moyen d'un encodeur convolutionnel récursif et systématique de la manière suivante (arithmétique modulo 2)
 - $e_0 = 0$ et $e_i = e_{i-1} + s_i \quad \forall i = 1, 2, \dots$
 - $x_{i,1} = s_i \quad \forall i = 1, 2, \dots$
 - $x_{i,2} = e_i \quad \forall i = 1, 2, \dots$
- les symboles $x_{i,1}$ et $x_{i,2}$ sont transmis successivement sur le canal,
- le canal produit un bruit binaire $n_{i,j}$ qui est ajouté au symboles $x_{i,j}$ pour former la suite de symboles de sortie : $y_{i,j} = x_{i,j} + n_{i,j} \pmod{2}$.

Dans le cadre du travail on se limitera à la simulation de la transmission de messages source de longueur 4.

On considérera différents cas de figure selon que la source et/ou le canal ont de la mémoire. Par exemple, la figure 2.1 montre le réseau bayésien correspondant au cas où la source et le canal ont de la mémoire. Dans ce réseau, on a créé une variable supplémentaire "s1s2s3s4" dont les 2^4 valeurs correspondent à la concaténation des 4 bits source. Par ailleurs, la variable e_0 n'est pas représentée explicitement car elle est constante.

Les fichiers de définition des différents réseaux bayésiens utilisés pour ce travail figurent dans le répertoire

/home/lwh/JBex/decodage/

Noter que les propriétés statistiques de la source S à l'entrée du canal ne sont pas fixées.

2.2.1 Source sans mémoire, bruit sans mémoire

Pour cet exemple il faut charger le fichier "simple.bif" dans JavaBayes. Fixer les propriétés de la source S à l'entrée du canal à celles de la source sans mémoire S_1 .

1. Analyse du système

- (a) Comment peut-on caractériser la source Y des symboles de sortie du canal ?
- (b) Comparer le débit du code (bit source/utilisation) avec la capacité du canal (Shannon/utilisation) et tirer les conclusions en ce qui concerne le taux d'erreur minimal réalisable sur ce canal avec ce débit.
- (c) Que peut on dire a priori sur le taux d'erreur théoriquement réalisable sur ce canal avec un tel code ?
- (d) Quelle est la distance de Hamming du code ?

2. Codage/décodage

Simuler la production par la source S d'une série de séquences s_1, s_2, s_3, s_4 ainsi que leur passage dans le canal. Relever les séquences de 8 bits obtenues en sortie.

A l'aide de l'option 'Observe' de Javabayes, fixer les variables $y_{1,1}, y_{1,2}, y_{2,1}, \dots, y_{4,2}$ aux valeurs obtenues au point précédent, et effectuer le décodage de chaque séquence obtenue par simulation selon les deux méthodes suivantes :

(a) DECODAGE BIT PAR BIT :

- i. Appliquer l'option 'Query' de Javabayes à chaque bit de la séquence s_1, s_2, s_3, s_4 afin de déterminer le bit source s_i le plus probable, pour $i = 1, \dots, 4$.

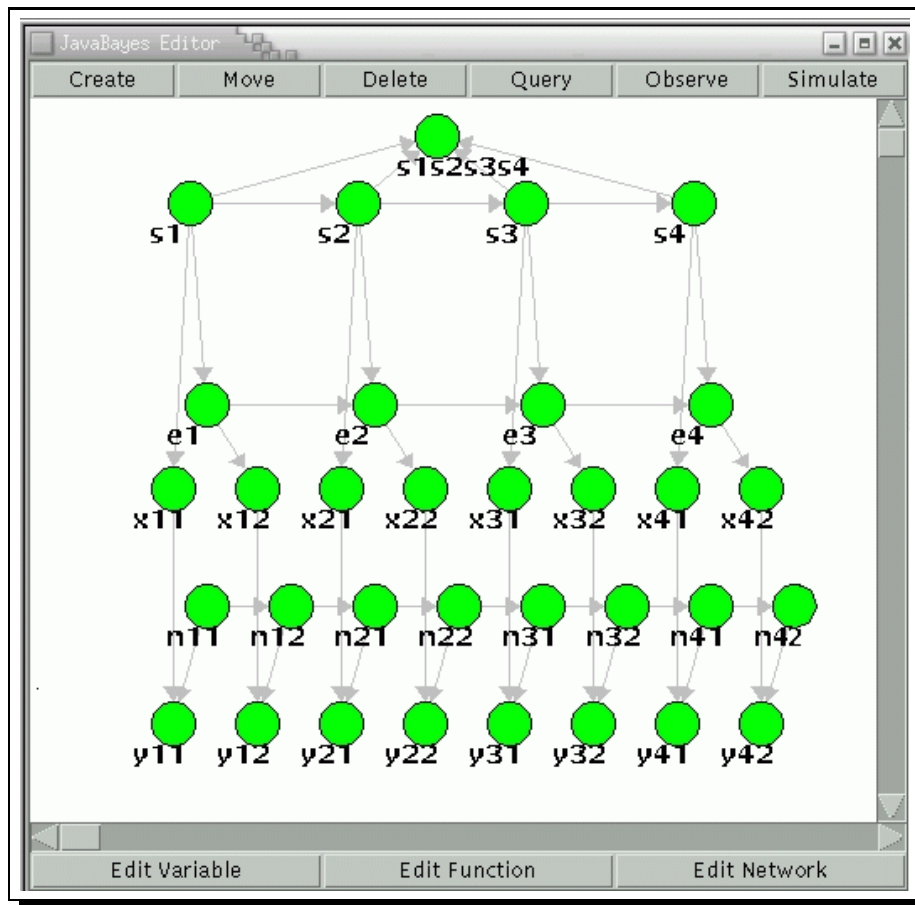


Figure 2.1. Réseau Bayésien pour un petit problème de codage/décodage de canal

- ii. Le codage augmente-t-il les chances de décodage correct pour toutes les séquences bruitées? En particulier, distinguer le cas d'erreurs simples, doubles ou de multiplicité supérieure. Illustrer par des exemples.
 - iii. A quel algorithme ce procédé de décodage correspond-il et quelle est la probabilité d'erreur minimisée par ce procédé ?
- (b) DECODAGE PAR MOT :
- (La variable artificielle "s1s2s3s4" a été ajoutée au modèle pour permettre un décodage par mot grâce à l'option 'Query' de Javabayes.)
- i. En quoi ce type de décodage par mot diffère-t-il du décodage par bit? Expliciter.
 - ii. Les séquences obtenues après décodage sont-elles identiques à celles obtenues lors du décodage bit par bit ? Vérifier en utilisant Javabayes et expliquer ce qui est observé.

2.2.2 Source sans mémoire, bruit de canal avec mémoire

Pour cet exemple il faut charger le fichier "BC_simple.bif" dans JavaBayes. Fixer les propriétés de la source S à l'entrée du canal à celles de la source sans mémoire S_1 .

1. Analyse du canal

Les symboles de bruit émis par le canal ne sont plus indépendants. Le processus de bruit forme une chaîne de Markov dont les probabilités de transition sont définies par (en utilisant la convention $n_{i,1} = b_{2i-1}$ et

$$n_{i,2} = b_{2i})$$

$$P(b_{i+1} = 1 | b_i = 1) = 0.6$$

$$P(b_{i+1} = 0 | b_i = 0) = 0.9$$

Les probabilités initiales sont $P(b_1 = 0) = 0.8$ et $P(b_1 = 1) = 0.2$.

- Par rapport au canal sans mémoire que peut-on dire en ce qui concerne le type des erreurs ?
- Si on alimente directement (sans encodeur) le canal par une source émettant des symboles équiprobables et indépendants, comment peut-on caractériser la sortie du canal (vue comme une source d'information) ?
- Déterminer la capacité du canal et en déduire le débit maximal de communication sans erreurs.

NB. La capacité d'un canal est définie en toute généralité par

$$C \triangleq \lim_{n \rightarrow \infty} \max_{P(\mathcal{X}^n)} \frac{I(\mathcal{X}^n; \mathcal{Y}^n)}{n} \quad (\text{Shannon/utilisation}) \quad (2.1)$$

où \mathcal{X}^n représente un message d'entrée de longueur n et \mathcal{Y}^n un message de sortie de longueur n .

2. Codage/décodage

Répéter les séries de simulations/décodage comme précédemment et commenter les résultats obtenus par rapport à ceux obtenus avec le canal sans mémoire.

2.2.3 Symboles source corrélés, bruit sans mémoire

Charger le fichier "SC_simple.bif" et fixer les propriétés de la source S à l'entrée du canal à celles de la source S_2 initialisée avec sa distribution stationnaire.

On demande

- si la source X qu'on observe en sortie du décodeur forme un processus de Markov,
- de déterminer l'entropie de cette source X , comparer avec l'entropie de la source S à l'entrée du canal, expliquer la différence observée (en Shannon/symbole binaire),
- de déterminer les entropies conditionnelles moyennes

$$H(\mathcal{Y}_{i,j} | \mathcal{X}_{i,j}) \quad (2.2)$$

$$H(\mathcal{Y}_{1,1}, \mathcal{Y}_{1,2}, \mathcal{Y}_{2,1}, \mathcal{Y}_{2,2}, \dots, \mathcal{Y}_{k,1}, \mathcal{Y}_{k,2} | \mathcal{X}_{1,1}, \mathcal{X}_{1,2}, \mathcal{X}_{2,1}, \mathcal{X}_{2,2}, \dots, \mathcal{X}_{k,1}, \mathcal{X}_{k,2}) \quad (2.3)$$

3 SIMULATION DE CHAÎNES DE MARKOV AVEC JAVABAYES

Certaines facilités ont été ajoutées au programme “JavaBayes” pour la modélisation et la simulation de sources de Markov de mémoire 1. Ces facilités sont décrites ci-dessous. La plupart de ces modifications sont accessibles via les menus “edition” et “simulation”.

3.1 SÉLECTION DE VARIABLES

Pour sélectionner une ou plusieurs variables, on utilisera le bouton “select” de la fenêtre “Javabayes editor” et on cliquera sur les variables à sélectionner. Pour désélectionner une variable, on cliquera à nouveau sur cette variable. L’option “unselect all” du menu “edition” permet de désélectionner toutes les variables sélectionnées.

3.2 CRÉATION DE VARIABLES IDENTITÉS

L’option “Create identity variable” du menu “edition” permet de créer une nouvelle variable qui est la concaténation des variables aléatoires sélectionnées (en rose sur le graphe). La nouvelle variable a pour valeurs toutes les concaténations possibles des valeurs des variables sélectionnées. La table de probabilités conditionnelles créée automatiquement par le programme et associée à cette nouvelle variable associe une probabilité de 1 lorsque la concaténation des valeurs des variables de conditionnement correspond exactement à la valeur de la nouvelle variable.

Cette option pourra par exemple être utilisée pour calculer la distribution de probabilité conjointe d’une séquence de variables d’une chaîne de Markov ou pour calculer une entropie conjointe.

3.3 CALCUL D’ENTROPIE CONDITIONNELLE

L’option “conditional entropy” qui se trouve dans le sous-menu “inference” du menu “options” permet d’activer le mode de calcul des entropies conditionnelles. Lorsqu’on utilise l’option “query” dans ce mode, le programme affiche l’entropie moyenne de la variable cliquée, conditionnée sur les variables sélectionnées et les observations. Pour rétablir le mode “query” habituel, on sélectionnera l’option “Posterior marginal” du même sous-menu. Attention, lors de vos expérimentations, tenez compte du fait que le calcul d’une entropie conditionnelle devient rapidement très lent lorsque le nombre de variables de conditionnement augmente.

3.4 SIMULATION DE VARIABLES

L'option "Simulate selected variables" du menu simulation effectue un tirage aléatoire des variables sélectionnées selon la distribution conjointe représentée par le réseau Bayésien et conditionnellement aux variables observées.

3.5 CRÉATION DE CHAÎNES DE MARKOV DE MÉMOIRE 1

L'option "create Markov chain" du menu "édition" permet de créer automatiquement un réseau bayésien à partir de la définition d'une chaîne de Markov. La création se fait en deux étapes. On détermine d'abord les paramètres de la chaîne de Markov, c'est-à-dire:

- le nombre d'états de la source,
- le nombre de pas de temps qui seront représentés par le réseau bayésien,
- un préfixe pour la création du nom des variables aléatoires représentant chaque état,
- un préfixe éventuel pour le nom des états de la chaîne de Markov qui, sinon, seront simplement les entiers de 0 à $M - 1$ si M est le nombre d'états.

Ensuite, on précise la matrice des probabilités de transition. En sélectionnant "dismiss" lors de cette étape, on utilisera des probabilités de transition uniforme par défaut. La distribution de probabilité initiale est, elle, toujours fixée à la distribution uniforme (les états sont équiprobables). Il appartient donc à l'utilisateur de la modifier en éditant la table de probabilités associée à la première variable de la chaîne de Markov.

Noter qu'une fois le réseau créé, toutes les options d'édition de JavaBayes restent applicables au réseau obtenu sans aucun changement. Par exemple, si vous modifiez la table des probabilités correspondant à une variable de la chaîne de Markov, le changement se sera pas propagé aux autres variables du réseau.

3.6 SIMULATION DE CHAÎNES DE MARKOV

L'option "simulate Markov chain" du menu simulation permet d'effectuer des simulations d'une chaîne de Markov dont on a préalablement sélectionné la première variable. Cette option affiche une nouvelle fenêtre dont les options sont décrites ci-dessous:

- Tables de probabilités: les tables de probabilités reprises au-dessus de la fenêtre sont les tables (initiale et de transition) correspondant à la chaîne de Markov sélectionnée. Ces tables peuvent être modifiées mais les modifications ne seront prises en compte qu'au sein du module de simulation et pas dans JavaBayes.
- "Pattern": un pattern est une séquence d'états de la chaîne de Markov (séparé par un espace ou une virgule). Le symbole '*' représente n'importe quel état.
- "Nb time steps" (T): le nombre de pas de temps qu'on désire simuler.
- "Nb realizations" (N): le nombre de réalisations utilisées pour calculer les statistiques d'ensemble.
- "Ensemble stats": calcule des statistiques d'ensemble sur le pattern. On génère N réalisations de la source, on vérifie l'apparition du pattern au début de chacune des réalisations et on renvoie la fréquence relative d'apparition du pattern parmi les N réalisations.
- "Temporal stats": calcule des statistiques temporelles sur le pattern. On génère une réalisation de longueur T , on vérifie l'apparition du pattern aux instants $t, t + 1, \dots, t + L - 1, \forall t = 0, \dots, T - L$ (où L est la longueur du pattern) et on renvoie la fréquence relative d'apparition du pattern sur les $T - L + 1$ pas de temps.
- "One realization": génère une réalisation de la chaîne de Markov de longueur T . Affiche cette réalisation dans la fenêtre "JavaBayes Console".