Utilisation des modèles de Markov cachés dans l'objectif de modéliser certaines séries chronologiques

Cédric Beaulac

Université du Québec à Montréal

27 novembre 2014

Table des matières

1 Introduction

1.1 Mise en situation

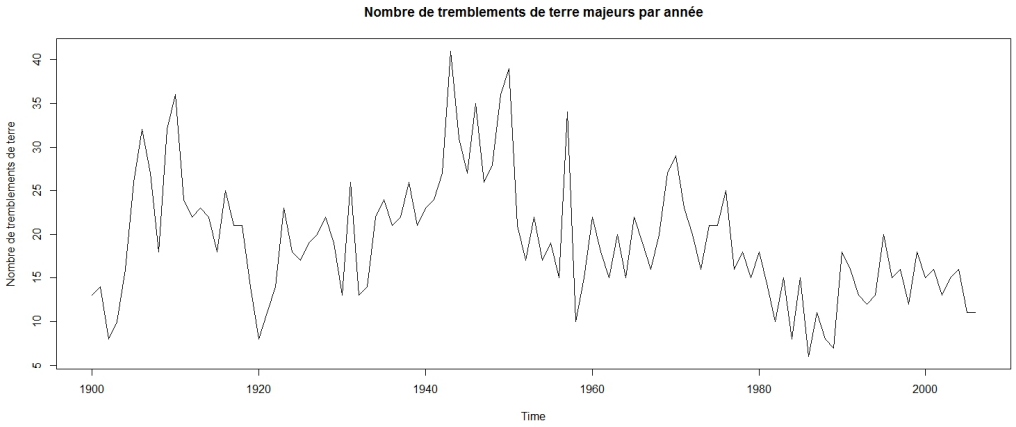
Supposons que nous analysons le nombre de parapluies vendus chaque jour par un certain kiosque au centre-ville. Comme il s'agit d'une variable aléatoire discrète qui modélise un nombre d'évènements dans un intervalle de temps, nous aurions tendance à considérer cette variable comme étant de Poisson. Par contre, il faudrait être dupe pour croire que ce processus est homogène puisque le vendeur de parapluies sera bien plus populaire lorsqu'il pleut que lorsque le soleil brille. Nous pourrions donc imaginer un modèle de Poisson avec une certaine moyenne lorsqu'il fait beau et un autre modèle de Poisson avec une moyenne plus élevée lorsqu'il pleut.

Supposons maintenant que la température d'aujourd'hui affecte celle de demain, ce qui est le cas, nous pourrions donc imaginer un modèle markovien, à deux états, soit il pleut ou soit il ne pleut pas. Nous serions alors face a une série chronologique qui est en fait un modèle de Markov caché, où les observations sont le nombre de parapluies vendus par jour, et la variable sous-jacente affectant la variable observée est la température extérieure.

Nous verrons désormais des exemples de véritables données ayant motivé l'utilisation des modèles de Markov cachés dans la modélisation des séries temporelles, ainsi que les justifications expliquant pourquoi cet outil est approprié dans certains cas.

1.2 Motivation

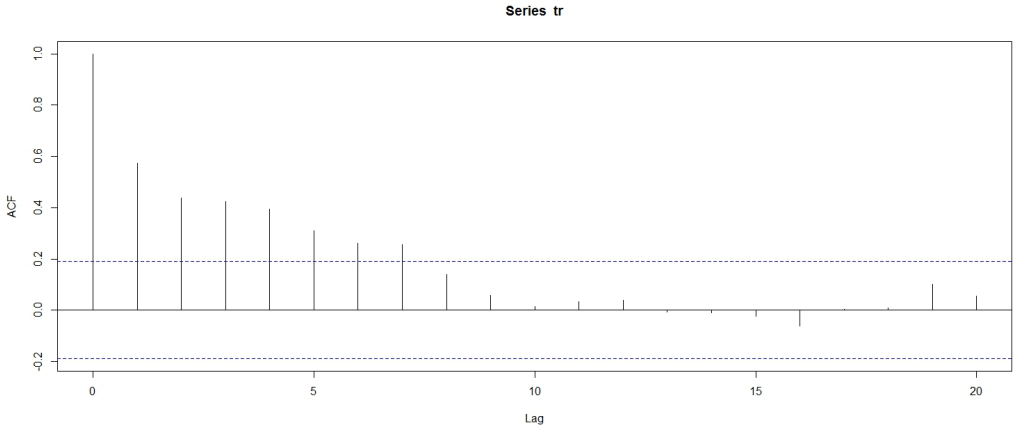
Nous traiterons d'un jeu de données concernant les tremblements de terre. Nous possédons le nombre de tremblements de terre majeurs, de magnitude supérieure à 7, par année, de 1900 à 2006. Ce jeu de données est à l'examen tout au long de l'ouvrage de Walter Zucchini [ZUC09], ma principale source de référence. Voici le graphique de cette série :



Les données, discrètes, représentant un comptage d'évènements peuvent être difficilement vues comme étant fonction d'un résidu normale, rendant difficile l'utilisation des processus ARMA standard. Ce type de variable serait plus intuitivement de Poisson. Néanmoins, une propriété connue des lois de Poisson est d'avoir une espérance égale à sa variance, hors ici nous avons une moyenne échantionnalle de 19.27 et une variance échantillonalle de 50.50.

Pour remedier à ce problème, Zucchini soulève l'idée d'utiliser les modèles mélangés indépendants (*independent mixture models*). Nous pourrions ici imaginer que nos données proviennent en fait de deux ou trois lois de Poisson différentes,

par exemple. Nous aurions de cette manière deux ou trois moyennes différentes, ce qui expliquerait la différence entre la variance et l'espérance. Dans ce type de modèle, des tirages indépendants d'une variable aléatoire sont faits pour déterminer laquelle de ces lois de Poisson est employée pour chaque observation. Bien que ce modèle est intéressant, il suppose l'indépendance entre chacune de nos observations, ce qui n'est pas le cas pour nos données. Observons le graphique des auto-corrélations :



Nous voyons clairement qu'il y a corrélation entre nos données. Il serait difficile de croire en l'indépendance de ceux-ci. Les modèles mixtes indépendants ne semblent pas être la solution ici. Malgré tout, il pourrait être intéressant de faire de plus amples recherches sur la modélisation des séries chronologiques à l'aide de ces modèles dans l'avenir.

L'idée est de percevoir la variable sous-jacente, celle qui décide quelle loi de Poisson employer, comme étant influencée par une séquence d'états passés. En voyant cette variable de la sorte, nous sommes de la sorte face à un modèle markovien. En supposant que la probabilité de choisir une certaine loi de Poisson ne

dépend que de celle qui a été utilisée précédémment, nous serons face à un modèle de Markov caché simple où la variable cachée est précisément cette variable markovienne dictant la loi de Poisson à utiliser. Ici, nos observations suivent une loi de Poisson, dont le paramètre est décidé par la chaîne de Markov sous-jacente, nous n'observons pas directemment les résultats de processus markoviens, mais bien une variable aléatoire sur ces résultats.

1.3 Plan du rapport

Nous commençerons avec une très courte revue des notions nécessaires par rapport aux modèles markoviens ainsi qu'aux probabilités de manière générale. Puis, nous enchaînerons avec une explication détaillée des modèles de Markov cachés ainsi que des divers algorithmes existant permettant de faire de l'inférence sur ceux-ci. Nous verrons par la suite comment utiliser ces derniers dans l'analyse des séries chronologiques. Par la suite, il sera question de la programmation de ceux-ci. Nous terminerons avec une brève analyse de quelques résultats ainsi que de potentielles améliorations et d'une voie exploratoire future.

2 Revue des notions nécessaire

Une importante base en probabilité est bien entendu nécessaire pour la compréhension de ce rapport de recherche. L'oeuvre de Ross [ROS10] est une bonne introduction aux probabilités si nécessaire.

De plus, une base concernant les chaînes de Markov est aussi essentielle à la compréhension de ce texte. Nous ferons ici une brève introduction de certaines caractéristiques. Un processus markovien discret est défini par des états et des probabilités de transitions. Notre variable aléatoire peut prendre comme valeurs chacun des états avec une certaine probabilité, mais cette probabilité n'est pas fixe, elle dépend de l'état dans lequel nous nous trouvons présentemment. La propriété markvienne s'écrit comme suit :

$$P(X_t = x|X_1 = x_1, X_2 = x_2, ..., X_{t-1} = x_{t-1}) = P(X_t = x|X_{t-1} = x_{t-1})$$

Prenons rapidement pour exemple un modèle voulant expliquer la température à chaque jour. Nous pourrions considérer *beau temps* et *pluvieux* comme température possible. Le probabilité qu'il fasse beau aujourd'hui dépenderait donc de la température qu'il y a fait hier. Finalement, il est possible de considérer un modèle où la température dépend des deux derniers jours. Il faudrait ici re-paramétriser notre modèle. Les états deviendraient donc la température des deux dernières journées. Il faut simplement garder en tête que l'état dans lequel nous nous trouvons contient toute l'information nécessaire quant aux calculs des probabilités de transition vers le prochain état.

Il est standard de stocker ces probabilités dans ce qu'on appelle une matrice de transition, disons A. L'élément à la ligne i et la colonne j représente la probabilité de passer de l'état i vers l'état j. Comme il s'agit donc de probabilité, chacune des

lignes doit sommer à 1. Grâce aux produit matriciels, il devient facile de calculer la probabilité de passer de l'état i vers l'état j en h étape, il s'agit de l'élément à la position (i, j) de la matrice A^h. Pour plus d'information sur les modèles de Markov, le livre de Ross est encore une fois une bonne source.

3 Modèles de Markov cachés

3.1 Présentation générale

Un modèle de Markov caché peut être décrit un peu rapidement comme étant une chaîne de Markov standard dont les états sont cachés. C'est-à-dire que nous avons un processsus Markovien dont les probabilités de transition ne dépendent que de l'état dans lequel nous nous trouvons. Cependant, les données que nous observons ne sont pas les états, mais bien une variable aléatoire sur les états. Au temps t, notre chaîne est à l'état i, mais nous n'obtenons pas l'état i comme observation, mais plutôt le résultat d'une variable aléatoire qui dépend de l'état dans lequel nous nous trouvons. Lorsque nous travaillons avec des modèles de Markov cachés, le défi est justement d'estimer les paramètres des lois d'observation et les probabilités de transition en n'ayant que nos observations comme données.

Il est important de noter que les observations peuvent être de n'importe quelle loi. Nous pourrions avoir des observations qui sont normales, alors les variables cachées pourraient influencer la moyenne, la variance ou bien les deux paramètres. La variable observée pourrait aussi ne suivre aucune loi de probabilité connue. Comme nos exemples concernaient cette dernière, et qu'il porterait à confusion de constamment changer de loi, nous allons, pour la totalité de ce rapport, considérer que la variable observée est de Poisson et que la variable markovienne cachée influence le paramètre λ.

Définissons quelques notations. Comme pour un processus de Markov standard, nous avons tout d'abord un vecteur d'états, par exemple $S = \{s_1, s_2, ..., s_n\}$, ainsi qu'un vecteur de probabilités initiales $\mu = \{\mu_1, \mu_2, ..., \mu_n\}$, ce dernier nous

informe de la probabilité de débuter notre processus markovien à chacun de nos n états. Nous aurons bien entendu une matrice A de taille n par n contenant les probabilités a_{ij} de transition de l'état i vers l'état j. Nous allons dénoter $Q = q_1, q_2....q_T$ la séquence des états visités lors des T transitions. Par contre, nous travaillons avec un modèle caché, alors, ce que nous aurons comme information, est une variable aléatoire associée à chacun des états, notons donc $X = x_1, x_2, ..., x_T$ la séquence d'observations obtenues de nos T états visités. Finalement, comme nous travaillons avec un modèle caché, nous devons avoir $b_i(x_t)$ qui est défini comme $P[X_t = x_t|q_t = S_i]$, la probabilité d'observer x_t sachant que la variable sous-jacente est à l'état s_i . Dans notre situation, supposons que nous avons un λ_i associé à chaque état s_i, donc $b_i(x_t) = P[X_t = x_t|q_t = S_i] = \frac{e^{\lambda_i}\lambda_i^{x_t}}{x_t!}$

Finalement, afin de faire de l'inférence sur les séries chronologique, nous aurons besoins de plusieurs algorithmes couramment utilisés à ses fins sur les modèles de Markov cachés. Grâce à ces algorithmes nous pourrons estimer la matrice de transition sous-jacente, mais aussi le paramètre λ des lois de Poisson associées à chaque état. Nous verrons dans la prochaine section l'algorithme $foward - backward$ et l'algorithme de $Baum - Welch$. Nous les verrons relativement rapidement considérant la lourdeur du rapport.

3.2 Algorithmes nécessaires à l'inférence

3.2.1 Algorithme *foward*

Définissons tout d'abord l'algorithme *foward*, celui-ci sert à calculer la valeur α. Définissons cette valeur. $\alpha_t(i) = P[x_1, x_2, ...x_t$ et $q_t = S_i]$. Il s'agit donc de la probabilité conjointe d'avoir observé la totalité des observations que nous avons

obtenus jusqu'au temps t et d'être à l'état sous-jacent s_i à ce temps.

Calculons donc $\alpha_{t+1}(j)$:

$$\begin{aligned}
\alpha_{t+1}(j) &= P(x_1, x_2, ...x_t, x_{t+1} \text{ et } q_{t+1} = S_j) \\
&= \sum_i^N P(x_1, x_2, ...x_t, x_{t+1} \text{ et } q_t = S_i \text{ et } q_{t+1} = S_j) \\
&= \sum_i^N P(x_{t+1} \text{ et } q_{t+1} = S_j | x_1, x_2, ...x_t \text{ et } q_t = S_i) P(x_1, x_2, ...x_t \text{ et } q_t = S_i) \\
&= \sum_i^N P(x_{t+1} \text{ et } q_{t+1} = S_j | q_t = S_i) \alpha_t(i) \\
&= \sum_i^N P(x_{t+1} | q_{t+1} = S_j) P(q_{t+1} = S_j | q_t = S_i) \alpha_t(i) \\
&= \sum_i^N b_j(x_{t+1}) a_{ij} \alpha_t(i).
\end{aligned} \tag{1}$$

Nous avons ici un algorithme récursif. Définissons la valeur initiale : $\alpha_1(i) = \mu_i b_i(x_1)$. Comme nous sommes face à des sommes de produit, nous pourrions être tenter de voir α_t comme étant un produit matriciel. Cette écriture offerte par Zucchnini est peut-être moins intuitive mais elle est beaucoup plus compacte et beaucoup plus simple à programmer. Définissons rapidement $P(x_t)$. Il s'agit d'une matrice diagonale, où les éléments $p_{i,i}(x_t)$ sont les probabilités d'avoir observé x_t sachant que nous étions à l'état i. Donc, soit A la matrice transition de la chaine sous-jacente, μ le vecteur de distribution initiale et $P(x_1) = diag(b_i(x_1))$ la matrice diagonale de probabilité d'observation nous constatons :

$$\begin{aligned}\alpha_1 &= [P(x_1, q_1 = S_1), P(x_1, q_1 = S_2), ..., P(x_1, q_1 = S_n)] \\ &= [\mu_1 b_1(x_1), \mu_2 b_2(x_1), ..., \mu_n b_n(x_1)] \\ &= \mu P(x_1).\end{aligned} \tag{2}$$

Notons que $\sum_i \alpha_1(i) = \sum_i P[X_1 = x_1, q_1 = S_i] = P[X_1 = x_1] = \mu P(x_1)1'$. Ou $1'$ est un vecteur colonne dont les éléments sont tous de valeurs 1.

De manière similaire, nous pourrions constater que : $\alpha_t = \mu P(x_1)AP(x_2)AP(x_3)...AP(x_t)$.

Finalment, la vraissemblance $L_T = P(X_1 = x_1, X_2 = x_2, ...X_T = x_T) = \sum_i \alpha_T(i) = \mu P(x_1)AP(x_2)AP(x_3)...AP(x_t)1'$.

3.2.2 Algorithme *backward*

Mettons maintenant au point l'algorithme *backward*. Définissons $\beta_t(i) = P[X_{t+1} = x_{t+1}, X_{t+2} = x_{t+2}, ..., X_T = x_T | q_t = S_i]$. Nous pourrions utiliser un développement similaire que pour α et nous obtiendrons $\beta_t = AP(x_{t+1})AP(x_{t+2})...AP(x_T)$. Il est possible de voir que $\forall t$, $L_T = P(X_1 = x_1, X_2 = x_2, ...X_T = x_T) = (P[x_1, x_2, ...x_t, q_t])(P[X_{t+1} = x_{t+1}, X_{t+2} = x_{t+2}, ..., X_T = x_T | q_t]) = \alpha_t \beta_t'$.

3.2.3 Algorithme *Baum-Welch*

L'algorithme de *Baum-Welch* est en fait une version adaptée aux modèles de Markov cachés de l'algorithme *Expectation–maximization (EM)*. Il s'agit donc d'un algorithme récursif qui mettera à jour nos paramètres lors de chaque itérations. Nous aurons ici besoin de deux propriétés. Tout d'abord observons que :

$$
\begin{aligned}
\gamma_t(i) &= P[q_t = S_i | x_1, x_2, ..., x_T,] \\
&= P[q_t = S_i | x_1, x_2, ..., x_t, x_{t+1}, ..., x_T,] \\
&= \frac{P[x_1, x_2, ..., x_t, x_{t+1}, ..., x_T, q_t = S_i]}{P[x_1, x_2, ..., x_T]} \\
&= \frac{P[x_1, x_2, ..., x_t, x_{t+1}, ..., x_T | q_t = S_i] P[q_t = S_i]}{P[x_1, x_2, ..., x_T]} \\
&= \frac{P[x_1, x_2, ..., x_t | q_t = S_i] P[q_t = S_i] P[x_{t+1}, x_{t+2}, ..., x_T | q_t = S_i]}{P[x_1, x_2, ..., x_T]} \\
&= \frac{P[x_1, x_2, ..., x_t, q_t = S_i] P[x_{t+1}, x_{t+2}, ..., x_T | q_t = S_i]}{P[x_1, x_2, ..., x_T]} \\
&= \alpha_t(i) \beta_t(i) / L_T.
\end{aligned}
\tag{3}
$$

Nous aurons ensuite besoin de calculer les probabilités de transition conditionnelle à nos observations :

$$
\begin{aligned}
\xi_t(i,j) &= P[q_{t-1} = S_i, q_t = S_j | x_1, x_2, ..., x_T,] \\
&= \frac{P[x_1, x_2, ..., x_T, q_{t-1} = S_i, q_t = S_j]}{P[x_1, x_2, ..., x_T]} \\
&= \frac{P[x_1, x_2, ..., x_{t-1}, q_{t-1} = S_i] P[q_t = S_j | q_{t-1} = S_i] P[x_t, x_{t+2}, ..., x_T | q_t = S_j]}{L_T} \\
&= \frac{P[x_1, x_2, ..., x_{t-1}, q_{t-1} = S_i] P[q_t = S_j | q_{t-1} = S_i] P[x_t | q_t = S_j] P[x_{t+1}, ..., x_T | q_t = S_j]}{L_T} \\
&= \alpha_{t-1}(i) a_{i,j} b_j(x_t) \beta_t(j) / L_T.
\end{aligned}
\tag{4}
$$

Avec ces deux outils en main, nous pourrons expliquer comment fonctionne l'algorithme de *Baum-Welch*. Rappelons qu'il s'agit d'un algorithme EM, nous

utilisons donc des techniques que vous connaissez peut-être. Nous allons observez une vraissemblance, avec des donnés connues et des données manquantes. L'étape E consiste à calculer l'espérance de la vraissemblance en utilisant l'espérance conditionnelle aux variables connues des variables manquantes. L'étape M consistera ensuite à maximiser le tout. Puis, nous mettrons notre modèle à jour et recommencerons à partie de l'étape E. Observons la vraissemblance des observations et des états comme suit :

$$\begin{aligned}
P[x_1, x_2, ..., x_T, q_1, q_2, ...q_T] &= \mu(q_1) b_{q_1}(x_1) a_{q_1,q_2} b_{q_2}(x_2) ... b_{q_T}(x_T) \\
&= \mu(q_1) \prod_{t=2}^{T} a_{q_{t-1},q_t} \prod_{t=1}^{T} b_t(x_t) \\
\Rightarrow \log(P[x_1, x_2, ..., x_T, q_1, q_2, ...q_T]) &= \log(\mu(q_1)) + \sum_{t=2}^{T} \log(a_{q_{t-1},q_t}) + \sum_{t=1}^{T} \log(b_{q_t}(x_t)) \\
&= \sum_{j=1}^{n} u_j(i) \log(\mu(q_1)) + \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{t=2}^{T} v_{i,j}(t) \log(a_{q_{t-1},q_t}) \\
&+ \sum_{j=1}^{n} \sum_{t=1}^{T} u_j(i) \log(b_{q_t}(x_t)).
\end{aligned} \tag{5}$$

Ou , $u_i(t) = 1$ ssi $q_t = s_i$ et $u_i(t) = 0$ sinon, et $v_{i,j}(t) = 1$ ssi $q_{t-1} = s_i$ et $q_t = s_j$, $v_{i,j}$ est égale à 0 sinon. L'étape E consistera à remplacer ces deux valeurs par leurs valeurs attendues sachant les observations :

$\hat{u}_i(t) = P[q_t = s_i | x_1, x_2, ..., x_T] = \gamma_t(i) = \alpha_t(i)\beta_t(i)/L_T.$

$\hat{v}_{i,j}(t) = P[q_{t-1} = s_i, q_t = s_j | x_1, x_2, ..., x_T] = \xi_t(i,j) = \alpha_{t-1}(i) a_{i,j} b_j(x_t) \beta_t(j)/L_T.$

Il ne nous reste maintenant que l'étape M, soit la maximisation. On observe que chaque terme de la log-vraissemblance peut être maximiser de manière disjointe. En maximisant chacun des termes, nous obtiendrons ces deux estimateurs :

$$\begin{aligned} \hat{\mu}(i) &= \gamma_1(i). \\ \hat{a}_{i,j} &= \frac{\sum_{t=2}^{T} \xi_t(i,j)}{\sum_{j}^{n} \sum_{t=2}^{T} \xi_t(i,j)} = \frac{\sum_{t=2}^{T} \xi_t(i,j)}{\sum_{t=2}^{T} \gamma_t(i)}. \end{aligned} \tag{6}$$

La troisième estimation varie selon la fonction d'observation, dans un contexte où celle-ci est de Poisson, nous obtenons : $\hat{\lambda}_i = \frac{\sum_{t=1}^{T} \gamma_t(i) x_i}{\sum_{t=1}^{T} = \gamma_t(i)}$.

Nous allons utiliser ces estimations pour calculer α et β à nouveau et l'algorithme convergera relativement rapidement vers les bonnes valeurs. Nous aurons une estimation de la matrice de transition ainsi que des divers paramètres λ. Nous verrons comment programmer ces algorithmes dans la section 5. Nous verrons aussi à quel point ils sont efficaces.

Bien qu'il ne s'agit que d'une petite parcelle de toute la théorie des modèles de Markov cachés, nous allons nous contenter de cela dans ce rapport pour analyser les séries chronologiques.

4 Modélisation des séries chronologiques

4.1 Lien d'autres modèles

Bien que nous venons de mettre au point un modèle permettant d'estimer les séries chronologiques à l'aide des modèles de Markov cachés, il est légitime de se demander s'il est possible de faire un lien entre ces modèles et des modèles ARMA plus classiques.

Les modèles TAR (*threshold autoregressive*) sont des modèles de type AR qui varient en fonction d'une variable aléatoire externe aux modèles. Brièvement, un modèle TAR est un modèle autoregressif pour lequel les divers paramètres varient en fonction d'une variable externe qui n'est pas nécessairement visible. Le modèle de Markov caché nous offre un peu plus de flexibilité quant à la variable observée et restreint la variable cachée à une variable markovienne. Ces deux modèles, quoique similaires, offrent une approche très différente dans l'inférence. Les TAR utilisent une structure similaire au processus ARMA classique lorsque nous savons le résultat de la variable cachée appelée le $threshold$ tandis que notre modélisation n'utilise en rien ce qui est couramment utilisé en analyse de séries temporelles.

Clemnts et Hans-Martin [CLE97] comparent justement l'efficacité des modèles *TAR* et des modèles de Markov cachés dans plusieurs aspects de l'inférence. Malgré que, selon lui, ces deux modèles décrivent de manière plus adéquate certains processus, dans le contexte de l'article l'article le PNB des États-Unis, ils ne sont tous deux pas plus efficaces en ce qui concerne la prédiction qu'un modèle ARMA classique. Malgré cette triste nouvelle, nous irons de l'avant avec nos

modèles de Markov cachés, cette section fut mise en place seulement pour démontrer que notre modélisation n'est pas complètement différente de tout ce qui a été fait historiquement en ce qui concerne les série temporelles ARMA.

4.2 Modélisation

4.2.1 Quand utiliser les modèles de Markov cachés ?

Il est important de se questionner à savoir quand devons-nous modéliser une série chronologique à l'aide des modèles de Markov cachés. Comme pour toute question statistique où il y a place à l'interprétation, il n'y a pas de réponse magique. Néanmoins, il y a certains cas où l'utilisation de ce modèle semble plus que logique, concrètement, les modèles où nous savons qu'il y une variable sous-jacente qui influence ce que nous voyons. L'exemple illustré en tout début de rapport, concernant la vente de parapluies, en est un, mais nous avons vu que le phénomène est aussi vrai en ce qui concerne les tremblements de terre. Plusieurs auteurs, Zhang [ZHA04] entre autre, essaient de modéliser certains indices boursiers à l'aide des modèles de Markov cachés en considérant la volatilité du marché comme étant une variable cachée, non-observable, mais qui influence la tendance des indices boursiers. Plusieurs autres données réelles peuvent être traitées de la sorte, les modèles des Markov cachés peuvent s'appliquer de manière réaliste à plusieurs sauces.

De plus, les modèles ARMA classiques réagissent mal lorsqu'il y arrive ce que nous appelons une rupture, soit un grand saut qui arrive à un certain temps et qui semble correspondre à un changement de moyenne, de tendance ou de modèle. Ces ruptures sont simples à interpréter dans une modélisation avec les chaînes

de Markov cachées. Nous percevons ces ruptures comme un changement d'état dans notre chaîne sous-jacente. Il serait donc intéressant d'essayer de modéliser certaines séries temporelles comportant des ruptures à l'aide des modèles markoviens cachés.

Finalement, quoi que cela peut sembler évident, il faut s'assurer que la variable sous-jacente soit bel et bien une variable aléatoire. Dans les séries cycliques, par exemple la vente de bikinis, on observe effectivement des moyennes différentes à travers le temps. Cette variable, la vente de bikinis, est en fait affectée par une variable sous-jacente, les saisons. Par contre, cette variable n'est pas aléatoire. Bien que l'été ne se fait pas toujours sentir à la même date, il succèdera toujours le printemps et les temps chauds arriveront toujours entre le début du mois de mai et la fin du mois de juin. La plupart des données cycliques sont effectivement influencées par une variable externe, mais, celle-ci n'est pas aléatoire. Les modèles de Markov cachés ne s'appliqueront donc pas ici.

4.2.2 Paramètres de la modélisation

Notre série chronologique, les données que nous possédons, jouent bien entendu le rôle des observations du modèle de Markov caché. Toutefois, le nombre d'état de la chaîne sous-jacente peut être difficile à déterminer. Lorsque nous connaissons la variable sous-jacente, par exemple la température, nous pouvons utiliser ces connaissances pour déterminer un nombre logique d'états cachés. Sinon, l'observation du graphique de la série temporelle peut nous donner une idée du nombre d'états nécessaires. Néanmoins, des tests statistiques robustes existent pour déterminer le nombre d'états idéaux, ceux-ci seront expliqués dans la prochaine section.

Un défi peut être de déterminer la loi d'observation. Malheureusement, peu d'outils sont construits expressement pour ses besoins. Nous devrons utiliser les méthodes usuelles de statistique, soit de supposer une certaine distribution et faire des tests. Lorsqu'il s'agit d'un nombre d'évènements dans un intervalle de temps, la loi de Poisson nous semble un choix logique. De plus, il est possible, en cas d'observation discrète, de considérer une variable observée non-paramétrique en supposant simplement que chaque état possède des probabilités différentes de générer chacune des observations possibles.

4.3 Vérification du modèle

Quelques outils sont couramment utilisés en régression, notamment pour vérifier la qualité d'un modèle. Des outils utilisant la vraissemblance et la log-vraissemblance qui sont couramment utilisés pourront s'appliquer ici. Comme nous le savons, à chaque fois que nous rajouterons un paramètre, nous augmenterons la vraissemblance. Certaines mesures ont été inventées pour considérer une punition lorsque nous ajoutons des paramètres. Ces outils ont pour objectif de réellement comparer des vraissemblances en considérant qu'il y a un prix à ajouter des variables. Nous voudrons le modèle qui obtient la meilleure vraissemblance par rapport au nombre de variables qu'il nécessite.

Rappelons que si la vraissemblance L augmente, la valeur $-2log(L)$ diminuera. Nous verrons deux statistiques, tout d'abord, le *Akaike Information Criterion* (*AIC*). Voici cette statistique :

$$AIC = -2\log(L) + 2p$$

Où p représente le nombre de paramètres dans une régression. Dans notre cas, si nous avons m états sous-jacents, nous estimons m paramètres λ, $m-1$ paramètres en ce qui a trait aux probabilités initiales, puis, $m*(m-1)$ paramètres pour la matrice de transition, bon pour un total de m^2+m-1 paramètres. Nous voulons ici utiliser comme nombre d'états celui qui produit l'AIC le plus petit.

Une autre statistique couramment utilisée en régression est le *Bayesian information criterion* (*BIC*). La voici :

$$BIC = -2\log(L) + p\log(n)$$

Dans ce cas, n représente le nombre d'observations que nous possédons dans notre échantillon. On veut encourager les bonnes vraissemblances qui ne nécessitent pas trop d'observations, pour deux vraissemblance similaires, l'une sera priorisée si celle-ci requière un échantillon plus petit que l'autre. Bien que ces deux statistiques se ressemblent, l'une incorpore la taille d'échantillon. De plus, le *BIC* favorise un nombre de paramètres souvent inférieur au *AIC*.

4.4 Prédiction

L'un des objectifs importants de l'inférence en série chronologique est de se doter d'outils de prévision. Nous introduirons rapidement des outils potentiels. Voyons ce que nous possèdons en travaillant avec les modèles de Markov cachés. Rappelons nous que si A est notre matrice de transition, ou $a_{i,j}$ est la probabilité de passé de l'état i vers l'état j en une transition, dans ce cas A^h contient les probabilités de passer de l'état i vers j en h étape. Rappelons-nous aussi que :

$$P[X_1 = x_1, X_2 = x_2, ..., X_T = x_T] = \mu P(x_1)AP(x_2)AP(x_3)...AP(x_T)1'$$

À l'aide de ces deux outils, calculons la probabilité d'une observation future connaissant un certain nombre d'observations :

$$\begin{aligned} P[X_{T+h} = x | X_1 = x_1, X_2 = x_2, ..., X_T = x_T] &= \frac{P[X_1 = x_1, X_2 = x_2, ..., X_T = x_T, X_{T+h} = x]}{P[X_1 = x_1, X_2 = x_2, ..., X_T = x_T]} \\ &= \frac{\mu P(x_1) A P(x_2) A P(x_3) ... A P(x_T) A^h P(x) 1'}{\mu P(x_1) A P(x_2) A P(x_3) ... A P(x_T) 1'} \\ &= \frac{\alpha_T A^h P(x) 1'}{\alpha_T 1'} \end{aligned} \tag{7}$$

Comme il nous est possible de cette manière de calculer ces probabilités pour tout horizon et pour toute valeur de x nous possédons alors tout le nécessaire pour faire des intervalles de prédictions pour tout horizon. Nous pourrions rajouter de plus amples détails quant à la prédiction, mais cette introduction suffit pour comprendre que les modèles de Markov cachés nous donne les outils nécessaires à la prévision. L'objectif personnel de notre recherche était de dégager les outils nécessaires à l'inférence et aux estimations des paramètres de notre modèle. Nous voulions néanmoins vous assurez que les outils nécessaires à la prévision existaient.

5 Programmation

Nous verrons ici les divers codes mis au point pour le programme R. Ces codes ne servent que lorsque la variable observée est de Poisson. Néanmoins, les modifications nécessaires pour adapter ceux-ci à d'autres modèles sont relativement simples. Nous nous sommes inspiré de nos connaissance sur le sujet ainsi que des codes fournis dans [ZUC09] pour mettre au point nos algorithmes.

5.1 Générateur

Nous allons tout d'abord programmer un générateur de modèle de Markov caché dont les observations sont de Poisson. Nous pourrons générer plusieurs variétés de modèles, avec peu ou beaucoup d'états par exemple, dans l'objectif de tester l'efficacité de nos algorithmes dans plusieurs situations. Voici notre générateur :

```
HMM.Pois.gen <- function(n,lam,P,pie) {
#n : Nombre d'observations simulées
#lam : Vecteur contenant les paramètres des variables de Poisson
#P : Matrice de transition de la variable sous-jacente
#pie : Vecteur de probabilités intiales/limites

 m <-  length(lam)
 lesetats <- 1:m
 etatvisite <- numeric(n)
 etatvisite [1] <- sample(lesetats ,1,prob=pie)
```

```
for (i in 2:n) {
 etatvisite[i]<-sample(lesetats ,1,prob=P[etatvisite[i-1] ,])
}

x <- rpois(n,lam=lam[etatvisite])
x
}
```

5.2 Algorithme *foward-backward*

C'est ici que nous ferons face au plus grand défi de programmation de ce rapport. Comme nous avons vu dans la section 3.2, les valeurs α et β sont des produits de probabilités et tenderont rapidement vers 0. Malgré que les ordinateurs peuvent manipuler de très petits nombres, leur capacité est limitée quant aux nombres de décimales et éventuellement ses valeurs seront arrondies à 0. Nous utiliserons donc une mise à l'échelle ainsi qu'une transformation logarithmique pour résoudre ce problème.

Discutons de la mise à l'échelle. Nous expliquerons ces techniques en faisant référence aux α, par contre, ce traitement de données est similaire en ce qui concerne les β. Posons $\hat{\alpha}$, notre variable mise à échelle. Nous allons tenter de faire en sorte que $\sum_i \hat{\alpha}_t(i) = 1$, nous ferons cela en divisant les α initiales par leur somme. Le tout est extrêmement bien expliquer dans [SHE08].

Initialisons nos variables de la sorte :

$$\begin{aligned} \ddot{\alpha}_1(i) &= \alpha_1(i) \\ c_1 &= \frac{1}{\sum_i \ddot{\alpha}_1(i)} \\ \hat{\alpha}_1(i) &= c_1 \ddot{\alpha}_1(i) \end{aligned} \tag{8}$$

Comme nous voulons qu'aucun résultat de nos calculs s'approchent trop près de 0, nous allons utiliser $\hat{\alpha}$ afin de calculer la totalité des itérations de la sorte :

$$\begin{aligned} \ddot{\alpha}_t(i) &= \sum_j^N b_i(x_t) a_{ji} \hat{\alpha}_{t-1}(j) \\ c_t &= \frac{1}{\sum_i \ddot{\alpha}_t(i)} \\ \hat{\alpha}_t(i) &= c_t \ddot{\alpha}_t(i) \end{aligned} \tag{9}$$

Grâce à cette méthode, nous aurons calculé la totalité de nos $\hat{\alpha}$ en s'assurant qu'aucun d'eux ne s'approchent trop de 0. Par contre, nos algorithmes nécessitent α et non $\hat{\alpha}$. Nous devrons alors trouver une manière de retrouver α. Analysons $\ddot{\alpha}_2(i)$ dans le but d'observer un résultat intéressant :

$$
\begin{aligned}
\hat{\alpha}_2(i) &= c_2 \ddot{\alpha}_2(i) \\
&= c_2 \sum_j^N b_2(x_t) a_{1,2} \hat{\alpha}_1(j) \\
&= c_2 \sum_j^N b_2(x_t) a_{1,2} c_1 \ddot{\alpha}_1(i) \\
&= c_2 c_1 \sum_j^N b_2(x_t) a_{1,2} \alpha_1(i) \\
&= c_2 c_1 \alpha_2(i) \\
&= \left(\prod_{k=1}^{2} c_k \right) \alpha_2(i)
\end{aligned}
\tag{10}
$$

De manière similaire, nous pourrions démontrer que : $\hat{\alpha}_t(i) = \left(\prod_{k=1}^{t} c_k \right) \alpha_t(i)$. Néanmoins, nous voulons tout de même travailler avec le logarithme de nos α pour s'assurer que ceux-ci ne soit pas arrondis à 0.

$$
\begin{aligned}
\hat{\alpha}_t(i) &= \left(\prod_{k=1}^{t} c_k \right) \alpha_t(i) \\
\Rightarrow \alpha_t(i) &= \left(\prod_{k=1}^{t} \frac{1}{c_k} \right) \hat{\alpha}_t(i) \\
\Rightarrow \log(\alpha_t(i)) &= \log \left(\left(\prod_{k=1}^{t} \frac{1}{c_k} \right) \hat{\alpha}_t(i) \right) \\
&= \sum_{k=1}^{t} \log(\frac{1}{c_k}) + \log(\hat{\alpha}_t(i))
\end{aligned}
\tag{11}
$$

Voyons finalement à quoi ressemble l'algorithme α en R en employant ces techniques :

```
HMM.Pois.Alpha2 <- function(x,lam,M,pie) {
#x : Les données
#lam : Les Lambdas ( paramètre de loi Poisson )
#M : Matrice de transition
#pie : Distribution initiale/stationnaire

n <- length(x)
m <- length(lam)

alpha <- matrix(data=rep(0,n*m),nrow=n,ncol=m)
proba <- outer(x,lam,dpois)

alph <- rep(0,m)
for ( i in 1:m ) {
alph[i] <- pie[i]*proba[1,i]
}
log1surc <- log(sum(alph))
alph <- alph/sum(alph)

alpha[1,] <-log(alph)+log1surc

for ( t in 2:n) {
alph <- alph%*%M*proba[t,]
```

```
log1surc <- log1surc + log(sum(alph))
alph <- alph/sum(alph)
alpha[t,] <- log(alph) + log1surc
}
alpha
}
```

Voyons l'algorithme β en R en employant ces techniques :

```
HMM.Pois.Beta2 <- function(x,lam,M,pie) {
#x : Les données
#lam : Les Lambdas ( paramètre de loi Poisson )
#M : Matrice de transition
#pie : Distribution initiale/stationnaire

n <- length(x)
m <- length(pie)

beta <- matrix(data=rep(0,n*m),nrow=n,ncol=m)
proba <- outer(x,lam,dpois)

beta[n,]<- rep(0,m)
bet <- rep(1/m,m)
log1surc <- log(sum(bet))

for ( t in 2:n) {
```

```
bet <- M %*% ( proba[n-t+2,]*bet)
beta[n-t+1,] <- log(bet) +log1surc
log1surc <- log1surc + log(sum(bet))
bet <- bet/sum(bet)
}
beta
}
```

5.3 Algorithme *Baum-Welch*

Nous verrons finalement comment fut implémenter l'algorithme de *Baum-Welch*. Comme les valeurs γ et ξ calculé en $3.2.3$ nécessitent la vraissemblance L_T et que celle-ci serait encore une fois arrondie à 0, nous devrons utiliser un autre petit truc. Celui-ci est mentionné dans [DUR98] et consiste à utiliser l'approximation suivante :

$$\log(p+q) = \log(p) + \log(1+\tfrac{q}{p}) = \log(p) + \log(1+\exp(\tilde{q}-\tilde{p}))$$

Ou $\tilde{q} = \log(q)$ et $\tilde{p} = \log(p)$. Nous voulons calculer la log-vraissemblance, ici nous dénoterons $\alpha_t(\tau) = max_i\{\alpha_t(i)\}$ et $c = \log(\alpha_t(\tau))$:

$$
\begin{aligned}
\log(L) &= \log\left(\sum_i \alpha_t(i)\right) \\
&= \log\left(\sum_{i\neq\tau} \alpha_t(i) + \alpha_t(\tau)\right) \\
&= \log(\alpha_t(\tau)) + \log\left(1 + \exp\left(\log(\sum_{i\neq\tau} \alpha_t(i)) - \log(\alpha_t(\tau))\right)\right) \\
&= c + \log\left(\exp(0) + \exp\left(\log(\sum_{i\neq\tau} \alpha_t(i)) - c\right)\right) \\
&= c + \log\left(\exp(0) + \exp\Big(\log(\sum_{i\neq\tau} \alpha_t(i)\Big)\exp(-c)\right) \\
&= c + \log\left(\exp(0) + \Big(\sum_{i\neq\tau} \alpha_t(i)\Big)\exp(-c)\right) \\
&= c + \log\left(\exp\Big(\log(\alpha_t(\tau)) - \log(\alpha_t(\tau))\Big) + \sum_{i\neq\tau} \exp(\log(\alpha_t(i)))\exp(-c)\right) \\
&= c + \log\left(\exp\Big(\log(\alpha_t(\tau)) - \log(\alpha_t(\tau))\Big) + \sum_{i\neq\tau} \exp\Big(\log(\alpha_t(i)) - \log(\alpha_t(\tau))\Big)\right) \\
&= c + \log\left(\sum_i \exp\Big(\log(\alpha_t(i)) - c\Big)\right)
\end{aligned}
\tag{12}
$$

À l'aide de cette technique nous permettant de calculer la log-vraissemblance, nous avons programmé l'algorithme de *Baum-Welch* comme suit :

```
HMM.Pois.BW3 <- function(x,lamini ,Mini ,pieini , iter) {
#x : Les données
#lamini : lambda initial de l'algorithme récursif
```

```
#Mini : Matrice de transition initiale
#pieini : Vecteur de probabilité initiale initiale
#iter : nombre d'itérations que fera l'algorithme



 m <- length(lamini)
 n <- length(x)


 for (k in 1:iter) {
nouvlam <- lamini
  nouvM <- Mini
  nouvpie <- pieini

  proba <- outer(x,nouvlam ,dpois ,log=TRUE)
  lalpha <- HMM.Pois.Alpha2(x,nouvlam,nouvM,nouvpie)
  lbeta <- HMM.Pois.Beta2(x,nouvlam,nouvM,nouvpie)
  p <- max(lalpha[n,])
   logl <- p+log(sum(exp(lalpha[n,]-p)))
for (i in 1:m) {
  for (j in 1:m) {
  nouvM[i,j] <- Mini[i,j]*sum(exp(lalpha[1:(n-1),i]+ proba[2:n,j]+lbet
  }
}
```

```
for ( i in 1:m) {
  nouvlam[i] <- sum(exp(lalpha[,i]+lbeta[,i]-logl)*x)/ sum(exp(lalpha[
  }

   pieini <- exp(lalpha[1,]+lbeta[1,]-logl) /sum(exp(lalpha[1,]+lbeta[

for ( j in 1:m) {
nouvM[j,] <- nouvM[j,]/ sum(nouvM[j,])
}
  lamini <- nouvlam
  Mini <- nouvM

 }
np <- m*m+m-1
AIC <- -2*(logl -np)
BIC <- -2*logl+np*log(n)
list(Mini,lamini,AIC,BIC)
 }
```

6 Simulations et résultats

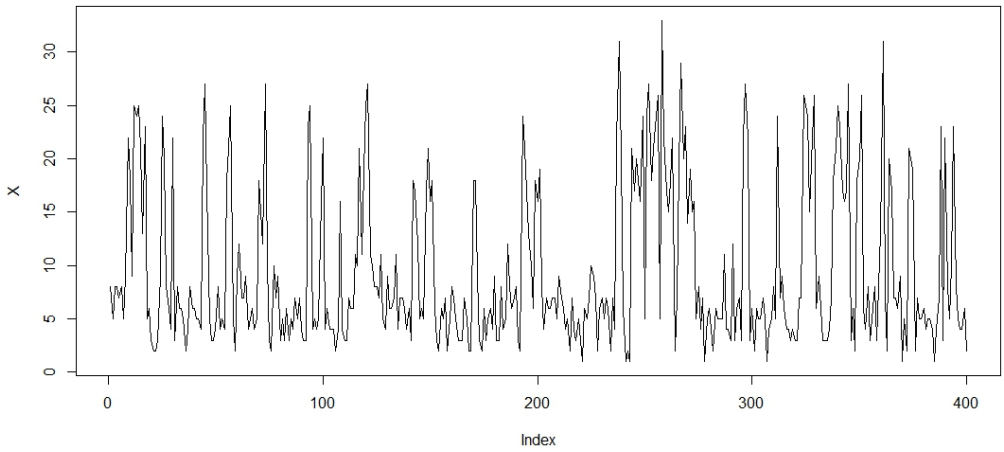
Nous verrons ici quelques simulations et quelques tentatives de retrouver les paramètres de notre processus. Nous utiliserons toujours 100 itérations dans l'objectif de démontrer que l'algorithme converge relativement rapidement et les probabilités initiales seront toujours distribuées de manière uniforme. De plus, nous ne montrerons pas les résultats de toutes nos expériences, par contre, nous en ferons un court résumé à la fin de cette section.

6.1 Modèle à deux paramètres : 5 et 20

Nous avons simulé 400 observations d'un modèle de Markov caché où la loi d'observation est de Poisson avec 5 et 20 comme paramètres. Voici la matrice de transition :

$$\begin{pmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{pmatrix}$$

Voici le graphique des observations :



Voici les estimations obtenues à l'aide de notre algorithme de *Baum-Welch* en R :

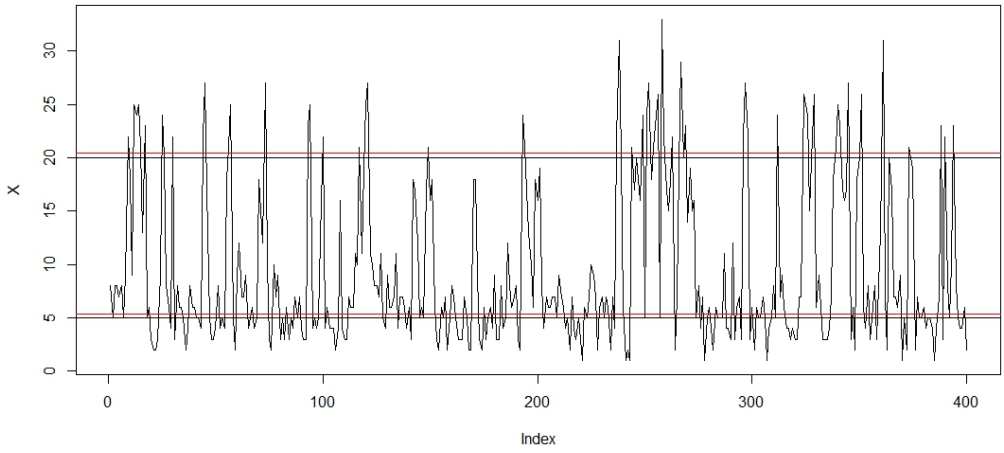
```
[[1]]
          [,1]      [,2]
[1,] 0.8898248 0.1101752
[2,] 0.2961164 0.7038836

[[2]]
[1]  5.331541 20.469968

[[3]]
[1] 2270.789

[[4]]
[1] 2290.747
```

Ces résultats sont plus que satisfaisants pour nous. Voici un graphique où nous avons tracé une droite noire aux vraies moyennes et rouge pour nos estimations :



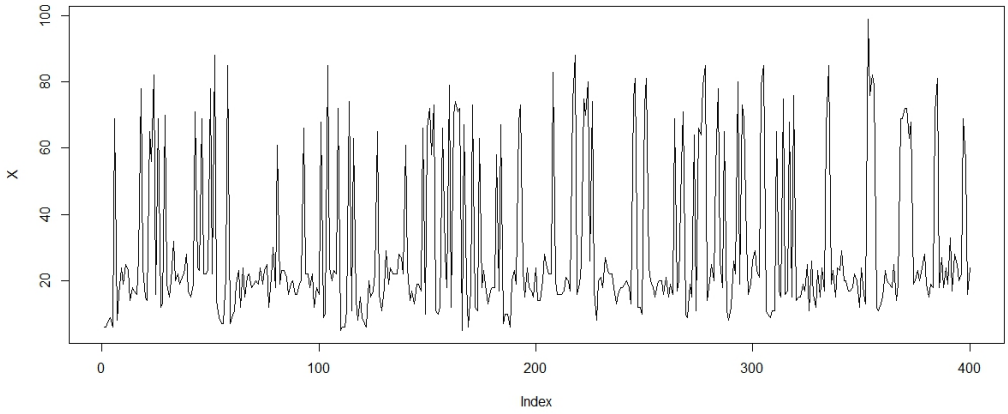
De plus, nous constatons que l'estimation de la matrice de transition est relativement bonne. Vérifions si la détection du modèle est adéquate en utilisant l'AIC. Notons que nous avions obtenue une valeur de 2270.789 pour un modèle à deux états cachés. Nous obtenons un AIC de 2273.685 en ce qui concerne un modèle à trois états. Cet outil de sélection de modèle nous inviterait donc à sélectionner celui à deux états comme nous le souhaiterions.

6.2 Modèle à trois paramètre : 10, 20 et 70

Nous avons simuler 400 observations d'un modèle de Markov cachées ou la loi d'observation est de Poisson avec 10, 20 et 70 comme paramètres. Voici la matrice de transition :

$$\begin{pmatrix} 0.6 & 0.2 & 0.2 \\ 0 & 0.9 & 0.1 \\ 0.3 & 0.3 & 0.4 \end{pmatrix}$$

Voici le graphique des observations :



Voici les estimations obtenues à l'aide de notre algorithme de *Baum-Welch* en R :

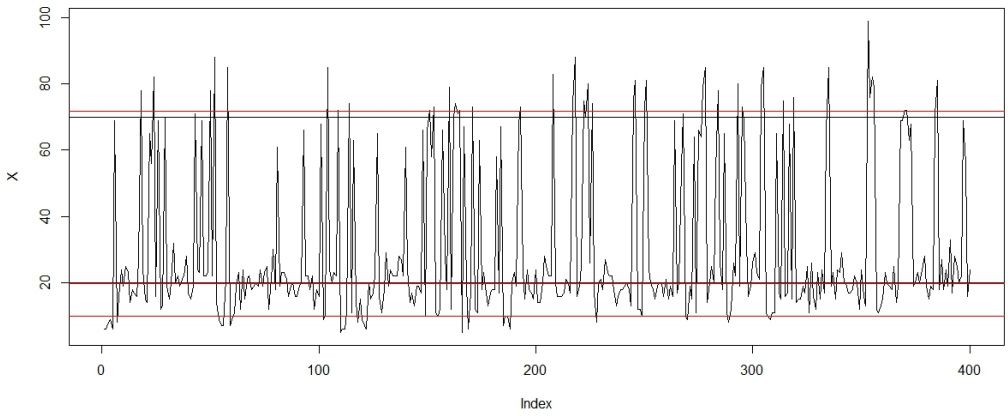
```
[[1]]
          [,1]      [,2]      [,3]
[1,] 0.6161251 0.1598583 0.2240166
[2,] 0.0000000 0.8412183 0.1587817
[3,] 0.2599950 0.3679120 0.3720931

[[2]]
[1] 10.00753 19.77410 71.88371
```

```
[[3]]
[1] 2874.262

[[4]]
[1] 2918.168
```

Ces résultats sont encore une fois satisfaisants pour nous. Voici le graphique où nous avons tracé une droite noire aux vraies moyennes et rouge pour nos estimations :



Comparons l'AIC obtenue avec celui pour un modèle à deux états et avec celui pour un modèle à quatre états. Nous avons ici obtenue une AIC de 2874.262 alors que pour un modèle à deux états nous aurions 3069.531 et que pour un modèle à quatre états nous aurions 2888.729. Ici, l'AIC favorise un modèle à trois états, ce qui est le bon choix.

6.3 Tremblement de terre

Pour nous assurer de la qualité de la programmation de nos algorithmes, nous avons utilisé ceux-ci sur les données réelles des tremblements de terre dont il est question dans [ZUC09]. En ce qui a trait aux AIC, nous avons obtenu 693.7978, 685.8776 et 694.7144 en ce qui concerne un modèle à deux, trois et quatre états respectivement. Nous favoriserions un modèle à trois états, tous comme les auteurs du livre en question.

De plus, en ce qui concerne l'estimation de la matrice de transition, les auteurs avaient obtenu :

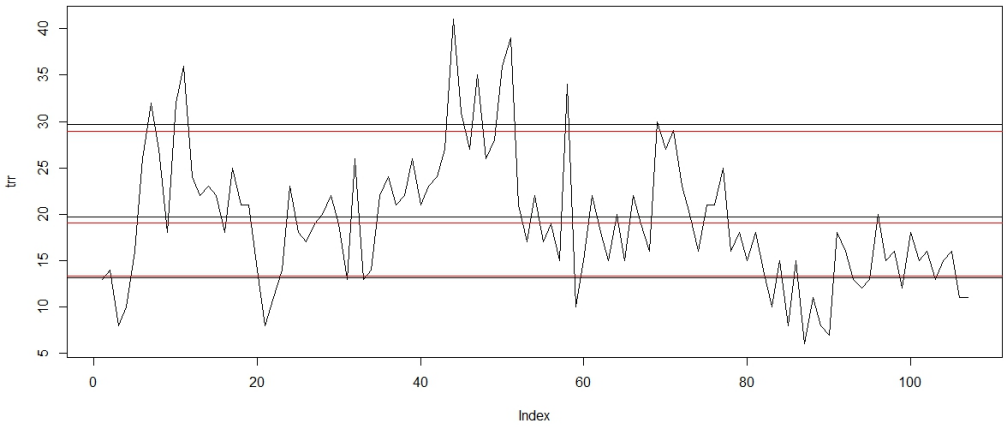
$$\begin{pmatrix} 0.955 & 0.024 & 0.021 \\ 0.050 & 0.899 & 0.051 \\ 0.000 & 0.197 & 0.803 \end{pmatrix}$$

Alors que nous avons obtenu :

$$\begin{pmatrix} 0.971 & 0.000 & 0.029 \\ 0.000 & 0.914 & 0.086 \\ 0.045 & 0.186 & 0.770 \end{pmatrix}$$

Bien que ces matrices ne soient pas exactement pareilles, les deux décrivent un comportement très similaire en ce qui a trait à la variable cachée. Comparons finalement le vecteur de paramètres de Poisson. Les auteurs ont obtenus $(13.146, 19.721, 29.714)$, alors que nous avons obtenu $(13.312, 19.032, 28.956)$. Encore une fois, nous observons des valeurs très similaires. Somme toute, nous sommes très satisfaits de nos codes. Voici un graphique des données observées, la

noire représente ici l'estimation des auteurs, et la rouge, la nôtre :



6.4 Observations

Nous ne ferons pas la liste des dizaines de simulations que nous avons faites. Nous laisserons le plaisir aux lecteurs du rapport de faire leurs propres expériences. Malgré tout, nous allons discuter de quelques observations.

6.4.1 Estimations des paramètres de Poisson

Parmi les deux estimations que nous voulons, celle des paramètres des lois de Poisson est la plus précise. Nous avons observer que le nombre d'états, donc de paramètres, ne semblent pas affecter la qualité de l'estimation en supposant que nous avons une certaine quantité d'observations. De plus, après une certaine quantité d'observations, l'ajout d'observations n'augmente plus réellement la qualité de celle-ci, l'algorithme n'as pas besoin d'un grand nombre d'observations pour bien estimer ces paramètres.

Bien sûr, le fait d'avoir deux états qui offrent presque le même taux de Poisson nuit à l'inférence. Par contre, d'un point de vue logique, peut-être que les deux états avec ce taux semblable ne forment en fait qu'un état et que nous avons considérer qu'il s'agissait de deux états sans raison valable.

6.4.2 Estimations de la matrice de transition

Cette estimation nous donne un bon aperçu de la réelle matrice de transition. Bien qu'elle ne soit pas extrêmement précise, l'estimation est suffisamment bonne pour que nous puissions avoir une bonne idée du comportement de la variable sous-jacente.

Comme dans un cas de modèle de Markov traditionnel, cette estimation gagne beaucoup en qualité en augmentant la quantité d'observations. Dans ce cas-ci, plus nous avons d'états à estimer plus nous avons besoin d'observation pour être efficace. Malgré tout, cette estimation est surtout utile pour comprendre de manière générale comment se comporte la variable cachée. Dans ce sens, on observe que la capacité de l'algorithme à détecter les éléments de la matrice de transition qui sont égaux à 0 ou à 1 est excellente.

6.4.3 Autres observations

L'algorithme converge très rapidement et de manière complètement arbitraire nous avons utilisé 1000 itérations et avons été satisfaits. Le calcul se fait relativement rapidement (moins de 30 secondes) et l'algorithme semble avoir convergé suffisamment à ce point. Souvenez-vous par contre que ce choix fut parfaitement arbitraire, il serait intéressant d'implémenter une variable qui nous fait sortir de l'algorithme lorsque nous avons assez convergé, comme les auteurs ont fait.

De plus, nous avons pris la décision de ne pas nous soucier de la distribution initiale du modèle markovien caché. Cette décision fut prise dans un souci d'économie de temps en considérant que selon nous, l'estimation des diverses moyennes de Poisson et de la matrice de transition sont les estimations qui sont intéressantes pour la compréhension de certains phénomènes.

7 Conclusion

Pour conclure, nous avons mis au point une manière de percevoir une série chronologique qui nous a permis de modéliser un jeu de données d'une méthode complètement différente. Les modèles de Markov cachés sont non seulement une manière logique de voir certains ensembles de données mais aussi un outil efficace en ce qui concerne l'inférence et la prédiction. Malgré que rien ne prouve pour l'instant, qu'ils sont plus efficaces que des modèles classiques, cette idée semble avoir un avenir.

Nous avons seulement travaillé avec des variables observées de Poisson, et les résultat sont satisfaisants. Il aurait été intéressant de travailler avec plusieurs types d'observations puisque l'une des forces de l'utilisation des modèles de Markov cachés est la flexibilité quant à la variable observée.

De plus, il serait intéressant d'essayer de voir comment utiliser des modèles de Markov cachés plus complexes, comme des modèles de Markov cachés cachés ou bien d'imaginer des modèles où la variable sous-jacente change la distribution de l'observation.

Finalement, il aurait pu être intéressant d'utiliser plus de données réelles dans des cas où nous croyons qu'il pourrait y avoir potentiellement une variable sous-jacente.

En conclusion, ce rapport de recherche fut une expérience enrichissante et j'espère que vous avez eu de l'intérêt et du plaisir à le lire.

8 Bibliographie

[ZUC09] ZUCCHINI, Walter & MACDONALD, Lain L. *Hidden Markov Models for Time Series : An Introduction Using R*. 2009, CRC Press.

[ROS10] ROSS, Sheldon M. *Introduction to probability models 10th edition*. 2010.

[CLE97] CLEMENTS, Michael P. & HANS-MARTIN, Krolzig. *A Comparison of the Forecast Performance of Markov-Switching and Threshold Autoregressive models of US GNP*. 1997.

[ZHA04] ZHANG, Yingjian *Prediction of financial time series with hidden Markov models*. 2004, Simon Fraser University

[SHE08] SHEN, Dawei *Some Mathematics for HMM*. 2008.

[DUR98] DURBIN, Richard & al. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids* . 1998