Elément de processus stochastique

Méthodes de Monte Carlo par chaînes de Markov - application à la cryptanalyse

Nora Folon, Julien L'hoest, Damien Seron

May 2018

1 Première partie : chaines de Markov pour la modélisation du langage et MCMC

1.1 Chaîne de Markov pour la modélisation du langage

1. On obtient la distribution de probabilité initiale π_0 d'une chaine de Markov modélisant cette séquence ainsi que la matrice de transition Q grâce à la méthode de vraisemblance.
   Pour calculer la matrice π_0, on utilise la formule suivante :

$$\pi_0(i) = P(X_0 = i) = \frac{\text{nombre de fois que } i \text{ apparaît}}{\text{nombre de lettre dans la séquence}}$$

   Ce qui nous donne dans ce cas :

$$\pi_0 = \begin{bmatrix} 0.3550 & 0.0750 & 0.2150 & 0.3550 \end{bmatrix}$$

   Pour trouver la matrice Q, on utilise la formule suivante :

$$[Q]_{i,j} = P(X_{t+1} = j | X_t = i) = \frac{\text{nombre de fois que } i \text{ précède } j}{\text{nombre de fois que } i \text{ précède une lettre}}$$

   On obtient alors :

$$Q = \begin{bmatrix} 0 & 0.0857 & 0.1000 & 0.8143 \\ 1.0000 & 0 & 0 & 0 \\ 0.6744 & 0 & 0 & 0.3256 \\ 0.3662 & 0.1268 & 0.5070 & 0 \end{bmatrix}$$

Nous pouvons donc construire le diagramme d'états de la chaîne de Markov correspondante à l'aide de la matrice Q :

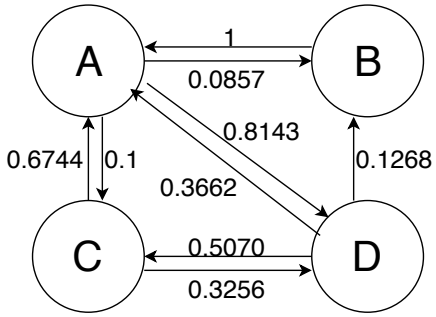

FIGURE 1 – Diagramme d'états

Pour la construction de la matrice de transition Q d'une chaîne de Markov à s états, la méthode de vraisemblance consiste à trouver les s^2 probabilités de transition qui sont définies comme tel :

Nous savons que pour une chaîne d'ordre 1 :

Cette équation, en réécrivant avec les probabilités de transition, devient :

La vraisemblance de la matrice Q s'écrit :

Avec n_{ij}, le nombre de fois que x_i est suivi de x_j dans `seq1` :

Le calcul se simplifie en passant en log :

2. Sur la figure 2, on remarque que les probabilités de chaque lettre tend vers une valeur constante, qui est la même pour les deux graphiques. En effet, quelque soit la lettre avec laquelle on démarre la séquence, la probabilité d'apparition de chaque lettre reste constante. Cela s'explique par le fait que notre matrice Q est représentative d'une chaîne de Markov ergodique. C'est à dire qu'elle est apériodique, irréductible et qu'elle a un espace d'états fini. Elle est irréductible car en démarrant de n'importe quelle lettre, on a les possibilités d'atteindre les autres et elle est apériodique

$$Q^t = \begin{bmatrix} 0,351759706172227 & 0,0753768126790780 & 0,216080454474387 & 0,356783026674308 \\ 0,351759706172227 & 0,0753768126790780 & 0,216080454474387 & 0,356783026674308 \\ 0,351759706172227 & 0,0753768126790780 & 0,216080454474387 & 0,356783026674308 \\ 0,351759706172227 & 0,0753768126790780 & 0,216080454474387 & 0,356783026674308 \end{bmatrix}$$

3. Une distribution de probabilité π_s est dite stationnaire lorsque $\pi_s = \pi_s \cdot Q^t$, i.e. lorsque la distribution se stabilise.
Comme on peut le constater à la question précédente, la matrice de transition

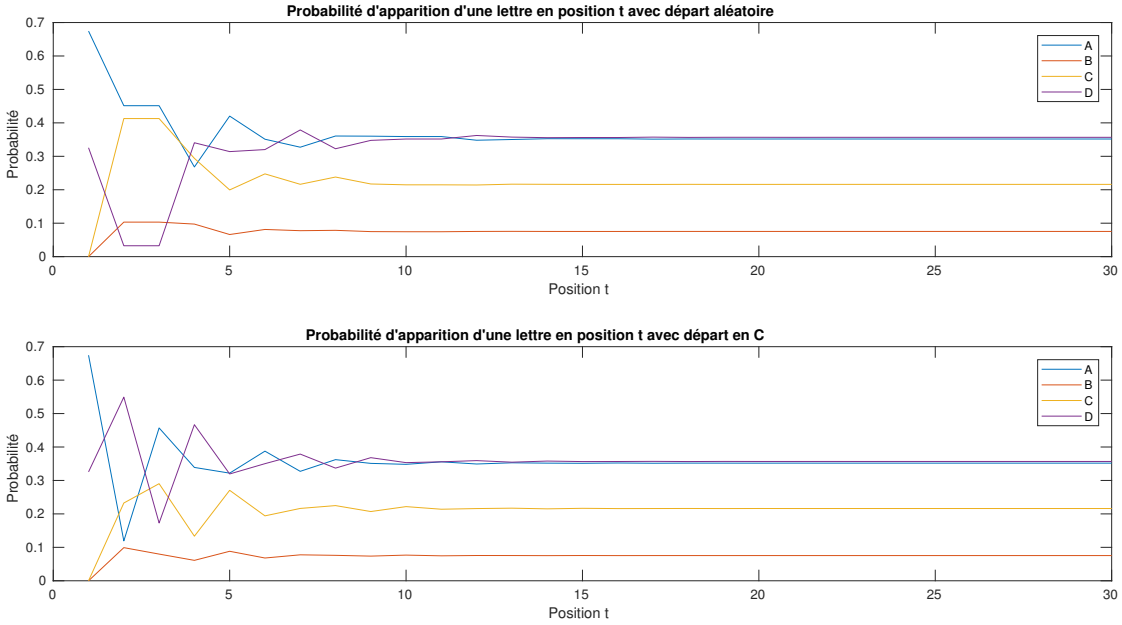

FIGURE 2 – Evolution des apparitions de chaque lettre (en pourcentage) avec départ aléatoire et avec départ en C

Q tend vers une valeur limite, Q^t. De là, il est aisé de trouver la distribution stationnaire π_∞ :

$$\pi_\infty = \begin{bmatrix} 0,351759706172227 & 0,0753768126790780 & 0,216080454474387 & 0,356783026674308 \end{bmatrix}$$

4. Comme on peut le voir à la Figure 3, les occurrences d'apparition des lettres rapportées à T convergent assez rapidement vers les valeurs de la distribution stationnaire π_∞. En effet, les transitions des valeurs se stabilisent après t = 200 transitions.
5. La distribution de probabilité de la chaine de Markov représentée ici est stationnaire c'est à dire que $\pi_S = \pi_S * Q$. Dans une chaine initialisée avec une distribution stationnaire la distribution reste inchangée.

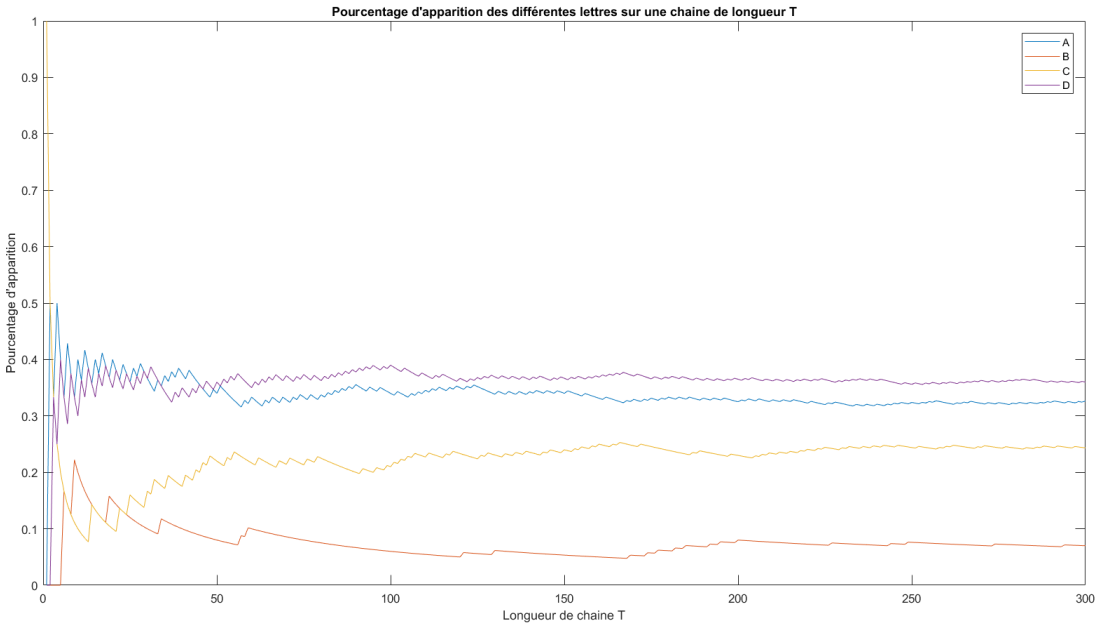

FIGURE 3 – Nombre d'apparition de lettres rapporté à la longueur de la réalisation

1.2 Algorithme MCMC

1. Soit une distribution de probabilité π_0 et une matrice de transition Q (NxN) qui satisfont les equations de balance détaillée :

$$\pi_0(i)[Q]_{i,j} = \pi_0(j)[Q]_{j,i} \tag{1}$$

On veut prouver que π_0 est stationnaire c'est à dire que $\pi_0 = \pi_0 * Q$. Détaillons le produit π_0 avec une colonne de Q.

$$\pi(i) = \pi_0(1)Q_{1,i} + \pi_0(2)Q_{2,i} + ... + \pi_0(N)Q_{N,i} \tag{2}$$

En utilisant les équations de balance détaillée on obtient :

$$\pi(i) = \pi_0(i)Q_{i,1} + \pi_0(i)Q_{i,2} + ... + \pi_0(i)Q_{i,N} = \pi_0(i) * (Q_{i,1} + ... + Q_{i,N}) \tag{3}$$

Or, la somme des éléments d'une ligne de la matrice de transition vaut 1. Ainsi,

$$\pi(i) = \pi_0(i) \tag{4}$$

Et donc en appliquant le même principe pour la multiplication entière on obtient $\pi_0 = \pi = \pi_0 * Q$ et π_0 est bien stationnaire.

2. On doit montrer que $\pi_0(i)[Q]_{i,j} = \pi_0(j)[Q]_{j,i} \iff p(X)P(Y|X) = p(Y)P(X|Y)$

$$\frac{C \cdot p_X(y^{(t)}) \cdot q(x^{(t-1}|y^{(t)})}{C \cdot p_X(x^{(t-1)}) \cdot q(y^{(t}|x^{(t-1)})} = \frac{C \cdot \pi(y)q(X|Y)}{C \cdot \pi(x)q(Y|X)} = \frac{\pi(y)q(X|Y)}{\pi(x)q(Y|X)}$$

Sachant que $\pi = c \cdot p_X$. On a alors :

$$\alpha = min\Big\{1, \frac{\pi(y)q(X|Y)}{\pi(x)q(Y|X)}\Big\}$$

et

$$Q_{x,y} = q(Y|X)\alpha(X,Y)$$

avec $q(Y|X)$ la probabilité que Y soit généré sur base de X et $\alpha(X,Y)$ la probabilité que Y soit sélectionné en tant que nouvelle état à la place de X.

- $\alpha < 1$ $\quad \pi(y)q(X|Y) > \pi(x)q(Y|X) \quad \longrightarrow \quad 1-\alpha$ % de chance de rejet
- $\alpha = 1$ $\quad \pi(y)q(X|Y) = \pi(x)q(Y|X) \quad \longrightarrow \quad$ acceptation
- $\alpha > 1$ $\quad \pi(y)q(X|Y) < \pi(x)q(Y|X) \quad \longrightarrow \quad$ acceptation

1^{er} **cas :**

$$\pi(y)q(X|Y) > \pi(x)q(Y|X), \qquad \text{alors}$$
$$\alpha(X,Y) = \frac{\pi(Y)q(X|Y)}{\pi(X)q(Y|X)}, \qquad \alpha(Y,X) = 1$$

Et donc

$$\begin{aligned}\pi(X)Q_{x,y} &= \pi(X)q(Y|X)\alpha(X,Y) = \pi(X)q(Y|X)\frac{\pi(Y)q(X|Y)}{\pi(X)q(Y|X)} \\ &= \pi(Y)q(X|Y) = \pi(Y)q(X|Y)\alpha(Y,X) = \pi(y)Q_{y,x}\end{aligned}$$

Finalement : $\pi(X)Q_{x,y} = \pi(Y)Q_{y,x}$

$2^{ème}$ **cas :**

$$\pi(y)q(X|Y) = \pi(x)q(Y|X), \qquad \text{alors}$$
$$\alpha(X,Y) = \alpha(Y,X) = 1$$

Et donc

$$\pi(X)Q_{x,y} = \pi(X)q(Y|X)\alpha(X,Y) = \pi(X)q(Y|X) = P(X,Y)$$
$$\pi(Y)Q_{y,x} = \pi(Y)q(X|Y)\alpha(Y,X) = \pi(Y)q(X|Y) = P(X,Y)$$

Finalement : $\pi(X)Q_{x,y} = \pi(Y)Q_{y,x}$

$3^{ème}$ **cas :**

$$\pi(y)q(X|Y) < \pi(x)q(Y|X), \qquad \text{alors}$$
$$\alpha(Y,X) = \frac{\pi(X)q(Y|X)}{\pi(Y)q(X|Y)}, \qquad \alpha(X,Y) = 1$$

Par un développement similaire au 1^{er} cas, on a finalement : $\pi(X)Q_{x,y} = \pi(Y)Q_{y,x}$

Grâce aux équations de balance détaillée, on sait qu'il existe une distribution stationnaire. Pour que l'algorithme de Metropolis-Hastings fonctionne, il faut également que la distribution stationnaire soit unique, i.e. que la chaîne de Markov soit ergodique.

2 Deuxième partie : décryptage d'une séquence codée

1. Afin de trouver la cardinalité de l'ensemble Θ de tous les codes possibles, il suffit de calculer la factorielle du nombre de lettres de l'alphabet. Ici, nous avons donc 40!
2. Afin de calculer la vraisemblance selon le modèle π_0, Q d'un texte T' non chiffré, il suffit de calculer le produit des probabilités d'avoir une lettre sachant la précédente, i.e. la probabilité d'observer le texte non chiffré sachant notre modèle.
   Ensuite, dans le but de trouver celle associée à un texte D résultant du chiffrement par un code θ donné, on cherche tout d'abord le texte non chiffré T à l'aide du code θ connu. En effet, on utilise l'inverse du code θ appliqué à Q pour trouver T. Enfin, on applique ensuite le même procédé qu'expliqué ci-dessus.
3. On identifie les dix codes pour lesquels $p_\theta(\theta)$ est égal à $\frac{1}{10}$. Pour chacun de ces codes, on calcule la vraisemblance du code déchiffré sachant π_0, Q. On prend ensuite le maximum de ces vraisemblances et on déchiffre avec le code correspondant.
4. Pour ce point nous avons choisi comme implémentation de tirer au hasard un code (qui est un mélange de l'alphabet grâce à la fonction `randperm`). A chaque itération du programme nous remplaçons le code par un nouveau code lui aussi tiré au hasard, ainsi q est indépendant du code actuel. La vraisemblance est calculée comme étant la somme des log_2 des probabilités plutôt que comme le produit des probabilités comme énoncé plus haut afin d'éviter de devoir jouer avec des nombres nécessitant un précision exagérée dès que la taille de texte grandit. Nous considérons qu'il y a convergence lorsque la vraisemblance du code trouvé est différente de -infini. En général cette méthode n'abouti a aucun résultat concluant.
5. Notre distribution de proposition q initiale est toujours aléatoire. Le fonctionnement du programme MATLAB est différent cette fois-ci. Tout d'abord, nous tirons un code au hasard. Ensuite, pendant un certain nombre d'itérations, nous tirons au hasard deux éléments de cette distribution que nous allons permuter. Nous calculons alors la vraisemblance du nouveau code, et la comparons avec la vraisemblance de l'ancien code. Si le nouveau code est plus vraisemblable que l'ancien, il est gardé, sinon il est rejeté avec une probabilité (1-α).

   Nous choisissons comme métrique le nombre de substitutions correctes dans

le code calculé. De plus, nous mesurons également l'évolution de la vraisemblance du code en parallèle de la métrique.

Afin de décoder le texte, nous allons évaluer notre métrique pour des segments du texte de plus en plus long. Pour l'instant, nous imposons un nombre d'itérations (nombre de fois que nous tirons une permutation envisageable des éléments du code) à 100000 de manière à pouvoir comparer l'exécution de chaque code en fonction de la longueur du texte. [1] Aussi, contrairement au point précédent, nous décidons de modifier légèrement notre calcul de vraisemblance.

Précédemment, lorsque qu'une transition intervenait et que la probabilité d'une telle transition était nulle, nous gardions la valeur de -Infini retournée par le logarithme, ce qui discréditait complètement le code et nous empêchait de commencer (ou continuer) par cette voie. A la place de discréditer complètement ce code, nous avons choisi de plutôt "pénaliser" une telle transition par une vraisemblance de -500 (en log2) au lieu de -infini ce qui permet de débuter et travailler à partir de n'importe quel code choisi aléatoirement. Il est bon de noter que dans une version antérieure du code nous ne remplacions pas la valeur de -infini et les résultats étaient moins concluants.

Afin de tester notre implémentation, nous avons choisi comme texte [2] :
"*i watch as gale pulls out his knife and slices the bread. he could be my brother. straight black hair, olive skin, we even have the same gray eyes. but we're not related, at least not closely. most of the families who work the mines resemble one another this way. that's why my mother and prim, with their light hair and blue eyes, always look out of place. they are. my mother's parents were part of the small merchant class that caters to officials, peacekeepers, and the occasional seam customer. they ran an apothecary shop in the nicer part of district 12. since almost no one can afford doctors, apothecaries are our healers. my father got to know my mother because on his hunts he would sometimes collect medicinal herbs and sell them to her shop to be brewed into remedies. she must have really loved him to leave her home for the seam. i try to remember that when all i can see is the woman who sat by, blank and unreachable, while her children turned to skin and bones. i try to forgive her for my father's sake. but to be honest, i'm not the forgiving type. gale spreads the bread slices with the soft goat cheese, carefully placing a basil leaf on each while i strip the bushes of their berries. we settle back in*

1. Nous introduisons un autre critère de convergence après, voir fin de la question 2.5
2. Suzanne Collins, *The Hunger games*, New York, Scholastic, 2008

a nook in the rocks. from this 10 place, we are invisible but have a clear view of the valley, which is teeming with summer life, greens to gather, roots to dig, fish iridescent in the sunlight. the day is glorious, with a blue sky and soft breeze. the food's wonderful, with the cheese seeping into the warm bread and the berries bursting in our mouths. everything would be perfect if this really was a holiday, if all the day off meant was roaming the mountains with gale, hunting for tonight's supper. but instead we have to be standing in the square at two o'clock waiting for the names to be called out."

Nous choisisons comme code de substitution :
'jlb." :[i,avfhq- ycsudo] ?tp) ;newrzxm !g('k'

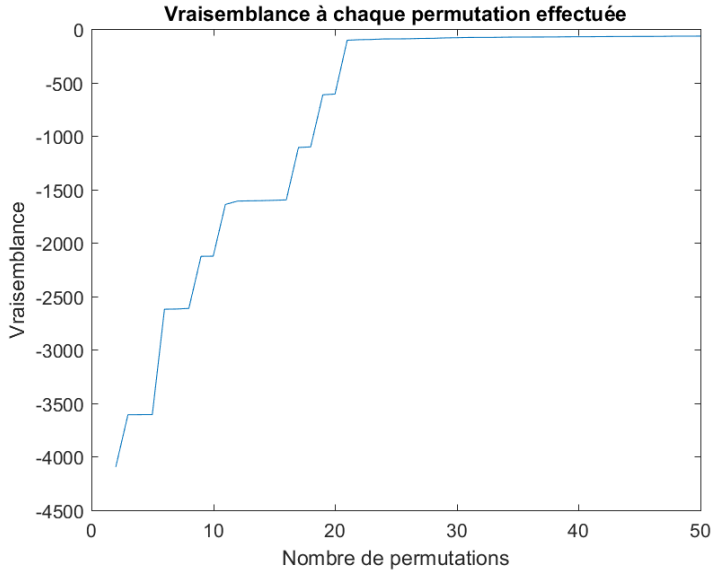

FIGURE 4 – Taille de texte 20

Taille 20 caractères
Comme on peut le voir à la Figure 4, la vraisemblance croît fortement pendant la première moitié des permutations, puis elle se stabilise pendant la deuxième. Il est bon de remarquer que malgré les 100000 itérations seulement 50 permutations ont eu lieu. Sur la Figure 5 on peut voir que le nombre de substitutions correctes ne dépasse pas 2 et peut aussi bien augmenter que diminuer sans qu'aucune véritable tendance ne transparaisse.

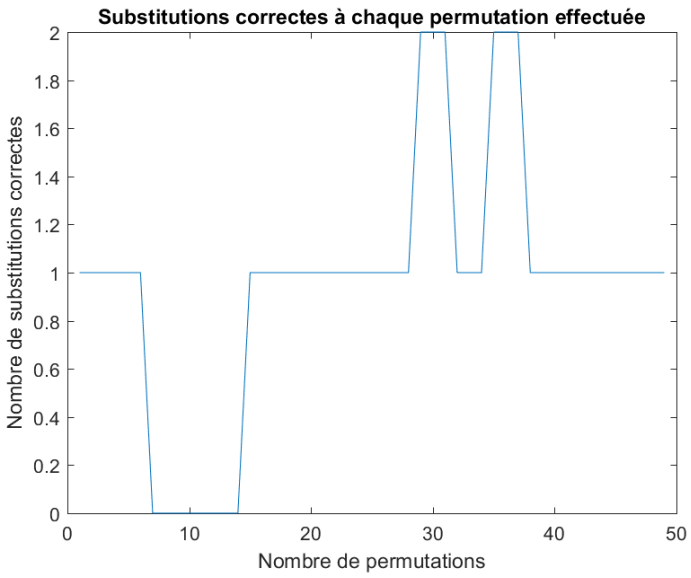

FIGURE 5 – Taille de texte 20

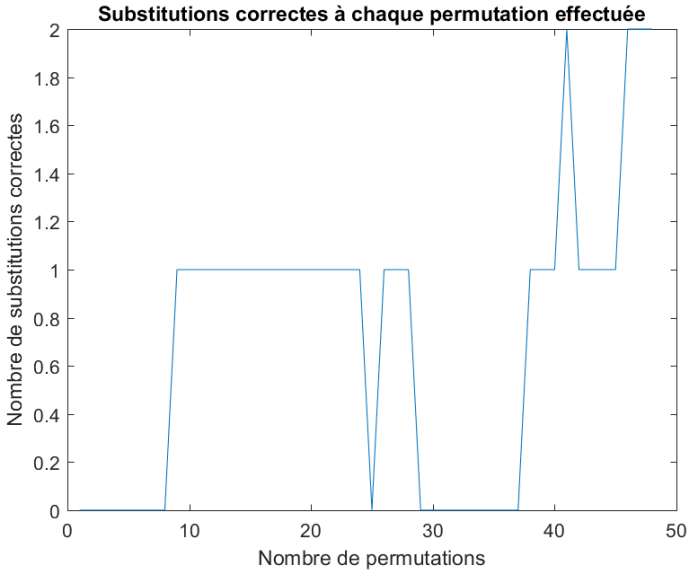

FIGURE 6 – Taille de texte 40

Taille 40 caractères
On peut voir sur la Figure 6 que le fait d'avoir doublé la taille du texte n'a pas eu d'impact significatif sur le nombre de substitutions correctes. De manière générale notre code Matlab semble en difficulté pour des échantillons de taille si petite (taille échantillon $\leq$ taille alphabet)

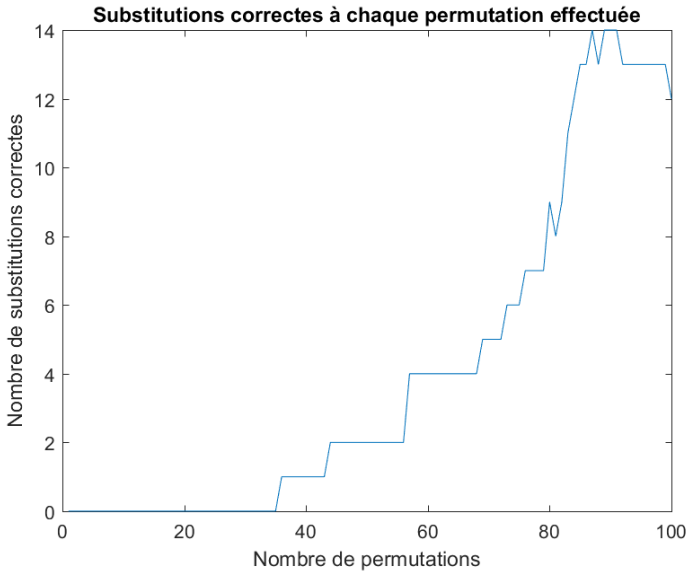

FIGURE 7 – Taille de texte 100

Taille 100 caractères
Comme l'illustre la Figure 7, la différence de résultat entre une taille de 40 et une taille de 100 est assez fulgurante. A la fin de l'exécution du programme, plus d'un quart des substitutions sont correctes et on peut aussi remarquer que le nombre de permutations effectuées est plus élevé, cela veut donc dire que plus d'occasions d'améliorer notre code se sont présentées.

Aussi, le nombre de substitutions correctes tend à augmenter de manière beaucoup plus rapide pour le dernier tiers des permutations. On peut peut aussi remarquer qu'il existe toujours des permutations qui font diminuer le nombre de substitutions correctes mais globalement leur proportion est moindre.

Taille 500 caractères
Le passage de 100 à 500 caractères confirme la tendance amorcée. Quelle

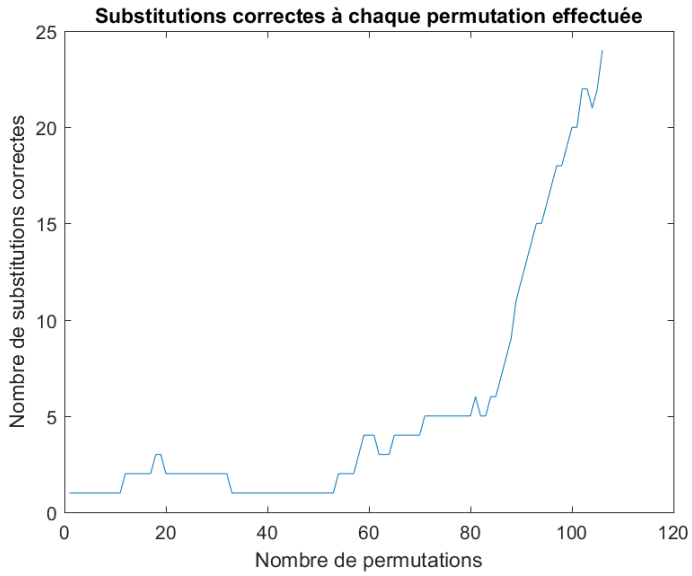

FIGURE 8 – Taille de texte 500

que soit la taille du texte étudié jusqu'ici, il semblerait que les 50 premières permutations ne soient pas très concluantes et qu'un certain travail de fond soit nécessaire pour permettre ensuite une évolution rapide du nombre de substitutions correctes.

En comparant la Figure 7 et la Figure 8, on peut remarquer que cette dernière présente des valeurs plus favorables après 100 permutations que la précédente après le même nombre de permutations.

Taille 1831 caractères (texte complet)

On remarque encore une fois sur la Figure 9 que aussi bien le nombre de permutations que le nombre de substitutions correctes est plus élevé à la fin de l'exécution que pour les longueurs inférieures de texte. Un autre fait intéressant est que, comparé aux tests précédents, les valeurs de substitutions correctes sont plus élevées dans les 50 premières permutations, atteignant 5 au bout de 50 permutations tandis que sur les Figures précédentes on pouvait bien observer une stagnation de la valeurs aux alentours de 1.

Maintenant que nous disposons des données pour l'évaluation du texte entier, il nous semblait opportun de comparer l'évolution du nombre de substitutions correctes et de la vraisemblance (respectivement Figure 9 et Figure 10).

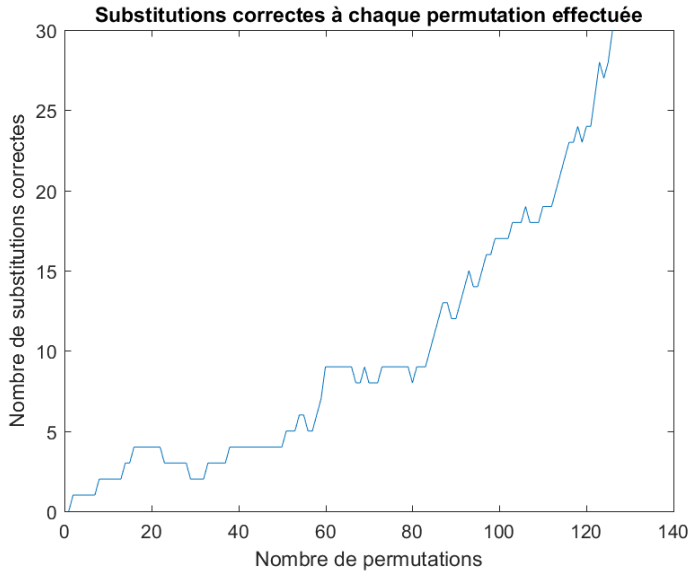

FIGURE 9 – Texte entier

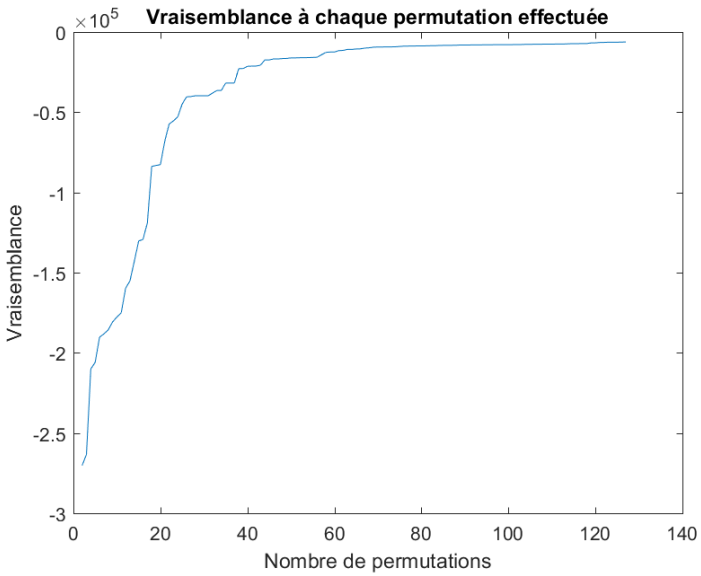

FIGURE 10 – Texte entier

L'évolution (croissance) des courbes semble se faire de manière inverse. Là où la vraisemblance se comporte comme un logarithme, l'autre graphe se comporte plutôt comme une exponentielle. La vraisemblance augmente de manière très rapide au début, ce qui est sûrement dû aux substitutions problématiques qui pénalisent fortement la valeur de la vraisemblance et qui sont par conséquent éliminées. Une fois que la vraisemblance est bien remontée et que les substitutions les plus problématiques ont été enlevées, la vraisemblance ne fera plus de grand bond mais chaque nouvelle permutation (effective) a plus de chance d'améliorer le nombre de substitutions correctes.

6. Notre choix de q n'est pas indépendant du code courant mais à chaque étape nous tirons de manière aléatoire et uniforme une permutation à effectuer sur le code. Ainsi, si une permutation est effectuée dans le code, la probabilité de proposer la permutation inverse à l'itération suivante est la même(q($x|y$) = q($y|x$)). Ce qui va empêcher de revenir au code précédent est le rapport des vraisemblances qui détermine α, le taux d'acceptation (le rapport des vraisemblances détermine complètement α car les probabilités de transitions de q s'annulent) . Nous avons fini par déchiffrer le texte donné :

"*will shared his unease. he had been four years on the wall. the first time he had been sent beyond, all the old stories had come rushing back, and his bowels had turned to water. he had laughed about it afterward. he was a veteran of a hundred rangings by now, and the endless dark wilderness that the southron called the haunted forest had no more terrors for him. until tonight. something was different tonight. there was an edge to this darkness that made his hackles rise. nine days they had been riding, north and northwest and then north again, farther and farther from the wall, hard on the track of a band of wildling raiders. each day had been worse than the day that had come before it. today was the worst of all. a cold wind was blowing out of the north, and it made the trees rustle like living things. all day, will had felt as though something were watching him, something cold and implacable that loved him not. gared had felt it too. will wanted nothing so much as to ride hellbent for the safety of the wall, but that was not a feeling to share with your commander. especially not a commander like this one.*"

Permutation = 'jby."]ri,anfoc-' :qsud !h ?lpwtve ;[zxm)g(k'

Nous avons aisément retrouvé la source à partir du texte [3]
Bien que notre programme actuel retrouve assez fréquemment le texte si le programme est lancé avec le texte entier, notre première découverte du texte

3. George R. R. Martin, *A Game of Thrones*, Bantam Books, 1996

est arrivée avec une version antérieure du code et sur laquelle nous avions travaillé seulement avec les 120 premiers caractères.

Critère de convergence Notre critère de convergence est une comparaison du nombre de caractères différents apparaissant dans le texte donné et du nombre de substitutions correctement devinées. Si notre nombre de substitutions correctes est au moins égal au nombre de caractères multiplié d'un coefficient nous considérons que la convergence est atteinte. Le coefficient est nécessaire car plus le nombre de caractères différent augmente plus il est difficile d'obtenir exactement toutes les substitutions correctes. En pratique dans le texte donné nous ne dépassons que rarement 30 substitutions correctes (le texte reste correct). Notre coefficient évolue généralement entre 0.7 et 0.8 mais nos temps de calculs sont assez variables (de quelques secondes à plusieurs minutes). En revanche, notre algorithme repose sur la chance de tomber sur des permutations favorables, et il arrive qu'aucune ne soit trouvée pendant un certain temps. Par ailleurs, notre critère de convergence ne convient pas vraiment pour les petits textes. Nous avons mis un garde fou lorsque 100 000 nouvelles itérations ont lieu sans améliorer le code afin d'éviter une attente trop longue (recommencer à partir d'un nouveau code se trouve souvent être plus efficace). Il est bon de noter que notre algorithme ne renvoie la phrase correcte qu'un fraction du temps, il peut être nécessaire de le lancer plussieurs fois.

3 Bonus

Nous avons eu deux idées de cryptage différents.

3.1

En premier lieu nous avons pensé à découper le texte en fragments de longueur N (le dernier fragment est possiblement de longueur $< N$). Chacun de ces fragments de texte subit un décalage de k positions vers la gauche ou vers la droite avec permutations circulaires. La valeur de k est tirée aléatoirement dans un domaine (par exemple $[-3; 3]$) pour chaque fragment. Ensuite, les fragments sont réassemblés en un texte. Notre programme est incapable de retrouver l'ancien texte à partir du nouveau et il faudrait un nouveau programme (adaptation impossible).

3.2

Le principe de ce codage repose aussi sur la substitution et fonctionne globalement comme le codage appliqué durant ce projet. La principale différence est l'introduction après chaque caractère du texte originel d'un autre caractère, à chaque fois tiré au hasard (distribution uniforme) dans l'alphabet. Par exemple si on remplace tout les 'a' par 'c' et tout les 'b' par des 'd' l'encodage de 'ab' pourrait tout aussi bien donner 'cxd!' que 'ccdh'. Tel quel notre programme serait incapable de décoder une telle séquence mais une simple adaptation lui faisant considérer uniquement un caractère sur deux le rendrait compatible. Il reste le doute concernant quelle moitié des lettres est correcte, il suffit de lancer le programme pour chaque moitié et de garder celui avec la plus grande vraisemblance.