3I005
Projet 3 : Analyse de séquences génomiques

Ariana Carnielli
Yasmine Ikhelif

Table des matières

1 Introduction

Les valeurs présentées dans ce document ont été calculées à l'aide des codes en Python dans le fichier `projet.py`, qui contient l'implémentation des fonctions demandées, et dans le Jupyter Notebook `Projet3.ipynb`, qui contient les appels de ces fonctions et l'affichage des résultats.

2 Préliminaires : données et lecture des fichiers

Question 1

Quatre fichiers ont été données pour ce projet. Leurs noms et les nombres de séquences et de nucléotides dans chacun d'entre eux sont présentés dans le tableau ci-dessous.

Nom du fichier	Nombre de séquences	Total de nucléotides
`regulatory_seq_PHO.fasta`	5	4000
`regulatory_seqs_GAL.fasta`	7	5608
`regulatory_seqs_MET.fasta`	9	7200
`yeast_s_cerevisae_genomic_chr1-4.fna`	4	2515853

Les chromosomes de *S. cerevisae* donnés dans le fichier `yeast_s_cerevisae_genomic_chr1-4.fna` contiennent ainsi 2515853 nucléotides.

Question 2

L'application de la fonction `nucleotide_frequency` à la liste des nucléotides de *S. cerevisae* donnés dans le fichier `yeast_s_cerevisae_genomic_chr1-4.fna` donne les fréquences présentées dans le tableau ci-dessous.

Nucléotide	Fréquence
A	0.29984
C	0.20110
G	0.20000
T	0.29907

Questions 3 et 4

Les fonctions `logproba` et `logprobafast` ont été codées dans `projet.py` et testées dans `Projet3.ipynb`.

3 Annotation des régions promoteurs

3.1 Description Empirique, préliminaires

Question 1

Les fonctions `code`, `inverse` et `comptage` ont été codées dans `projet.py` et testées dans `Projet3.ipynb`.

Question 2

On fixe un mot $w = w_0 w_1 \cdots w_{k-1}$ de longueur k et on considère une séquence aléatoire de longueur ℓ, $S_0 S_1 \cdots S_{\ell-1}$ avec $\ell \geq k$. On suppose que les variables aléatoires $S_0, \ldots, S_{\ell-1}$ correspondant à chaque lettre sont indépendantes et identiquement distribuées (leur loi étant estimée par la fonction donnée `nucleotide_frequency`). Soit X_i la variable aléatoire Bernoulli qui vaut 1 si et seulement si le mot w apparait dans S à la case i, c'est-à-dire $S_i = w_0$, $S_{i+1} = w_1$, $\ldots$, $S_{i+k-1} = w_{k-1}$. Remarquons que cela n'est possible que si $i + k - 1 \leq \ell - 1$, donc $i \leq \ell - k$. Comme les variables S_j sont indépendantes, on a

$$P(X_i = 1) = P(S_i = w_0) P(S_{i+1} = w_1) \cdots P(S_{i+k-1} = w_{k-1}).$$

De plus, comme les variables S_j sont identiquement distribuées, $P(X_i = 1)$ ne dépend pas de i (mais dépend de w). Notons $p = P(X_i = 1)$. Remarquons que p n'est rien d'autre que l'exponentielle de la valeur rendue par la fonction `logproba` lorsque son argument `liste_entiers` vaut w et son argument `m` contient la loi des S_j.

Soit X la variable aléatoire donnant le nombre d'occurrences de w dans S. On a alors

$$X = \sum_{i=0}^{\ell-k} X_i.$$

On remarque que, sauf dans le cas $k = 1$, les variables X_i ne sont pas indépendantes, donc X n'est pas forcément une variable aléatoire binomiale. Cependant, comme l'espérance est linéaire et $\mathbb{E}(X_i) = P(X_i = 1) = p$ car X_i suit une loi de Bernoulli, on peut calculer l'espérance de X par

$$\mathbb{E}(X) = \sum_{i=0}^{\ell-k} \mathbb{E}(X_i) = \sum_{i=0}^{\ell-k} p = p(\ell - k + 1).$$

Cette formule a été implémentée dans la fonction `comptage_attendu` codée dans le fichier `projet.py` et testée dans le fichier `Projet3.ipynb`.

Question 3

Les graphiques, obtenus dans le fichier `Projet3.ipynb`, sont données dans la Figure 1. On y remarque qu'il y a un écart important entre les nombres d'occurrence attendus et observés, qui peut être vu par la distance entre les points tracés et la diagonale, représentée en gris sur les figures. Cet écart est d'autant plus important que la longueur du mot recherché est grand. Ainsi, le modèle utilisé jusqu'à présent, qui suppose que deux lettres à des positions différentes sont des variables aléatoires indépendantes, ne permet pas d'expliquer les observations des occurrences des mots. Certains mots sont beaucoup plus fréquents que d'autres et cela ne peut pas être expliqué simplement par les fréquences de leurs lettres.

3.2 Simulation de séquences aléatoires

Question 1

La fonction `simule_sequence` a été codée dans `projet.py` et testée dans `Projet3.ipynb`. Pour la création de la séquence, on suppose que les lettres à des positions différentes sont indépendantes.

Question 2

On a simulé des séquences de grande longueur et tracé le graphique avec les nombres attendus et observées de chaque mot de longueurs k données. Les résultats pour une séquence de taille 1000000 et $k = 6$ sont donnés dans la Figure 2. On y observe que les points sont assez proches de la diagonale, comme attendu, car le calcul du nombre d'occurrences attendu se base sur l'indépendance des lettres de la séquence, utilisée pour la création de la séquence aléatoire. Ces points sont d'autant plus proches de la diagonale que la taille de la séquence générée est grande mais leur variance, et donc leur dispersion autour de la diagonale, augmente rapidement avec k.

Question 3

La fonction `proba_empirique` calculant la probabilité empirique d'un mot a été codée dans `projet.py` et testée dans `Projet3.ipynb`. Elle prend en argument

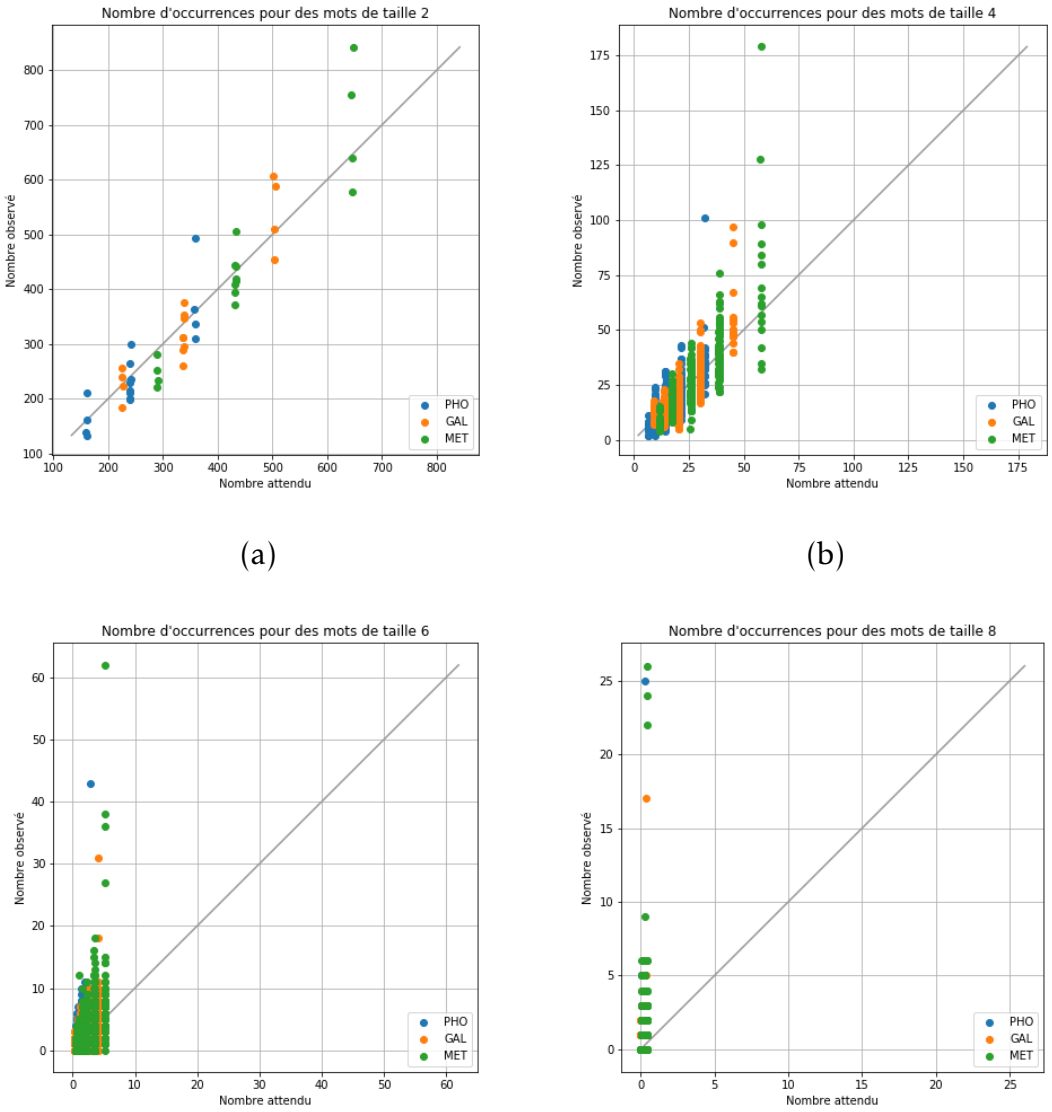

FIGURE 1 – Nombre d'occurrences attendu et observé pour (a) $k = 2$, (b) $k = 4$, (c) $k = 6$ et (d) $k = 8$.

un mot, la taille `lg` de la séquence à simuler, les probabilités de chaque nucléotide et un nombre de simulations à faire et renvoie un dictionnaire tel que l'élément de clé `i` contient la probabilité estimée que ce mot apparaisse exactement `i` fois dans une séquence de taille `lg`.

Question 4

La Figure 3 donne les histogrammes avec les probabilités estimées du nombre d'occurrences des mots ATCTGC, ATATAT, TTTAAA et AAAAAA pour des suites de longueur 10000 en faisant 1000 simulations pour chaque mot. Le code pour créer ces figures est donné dans le fichier `Projet3.ipynb`.

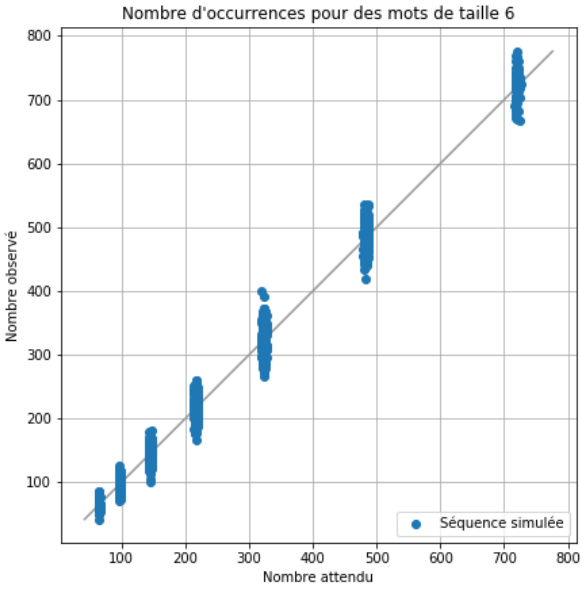

FIGURE 2 – Nombre d'occurrences attendu et observé pour une séquence de taille 1000000 générée aléatoirement.

On observe dans la figure que les probabilités d'occurrence dépendent du mot en question. Cela dépend des lettres qui composent ce mot, car A et T sont plus fréquentes que C et G, ce qui explique la différence de la Figure 3(a) avec les trois autres, mais les probabilités dépendent aussi du mot en soi. En effet, certains mots peuvent se chevaucher plus facilement, en on espère ainsi avoir des nombres d'occurrence plus grands de ces mots. Par exemple, en supposant les lettres A et T comme équiprobables, les séquences à 12 lettres ATATATATATAT, TTTAAATTTAAA et AAAAAAAAAAAA ont toutes la même probabilité d'apparaitre, mais la première donne lieu à 4 occurrences du mot ATATAT, la deuxième donne lieu à 2 occurrences du mot TTTAAA et la troisième donne lieu à 7 occurrences du mot AAAAAA, cette dernière étant la séquence qui s'auto-chevauche le plus facilement. En particulier, dans une séquence de longueur ℓ avec ℓ un multiple de 6, le nombre maximal d'occurrences du mot AAAAAA est $\ell - 6 + 1$, le nombre maximal d'occurrences de ATATAT est $\frac{\ell}{2} - 2$ et le nombre maximal d'occurrences de TTTAAA est $\frac{\ell}{6}$, ce qui indique déjà que les lois de probabilité du nombre d'occurrences de ces mots sont différents car leurs supports sont différents.

Question 5

Fixons un mot w et deux entiers n et ℓ. Soit N la variable aléatoire représentant le nombre d'occurrences de w dans des séquences aléatoires de longueur ℓ. Pour estimer $p = P(N = n)$, on peut considérer que p est le paramètre d'une variable aléatoire de loi Bernoulli qui vaut 1 si $N = n$ et 0 sinon. On peut alors utiliser

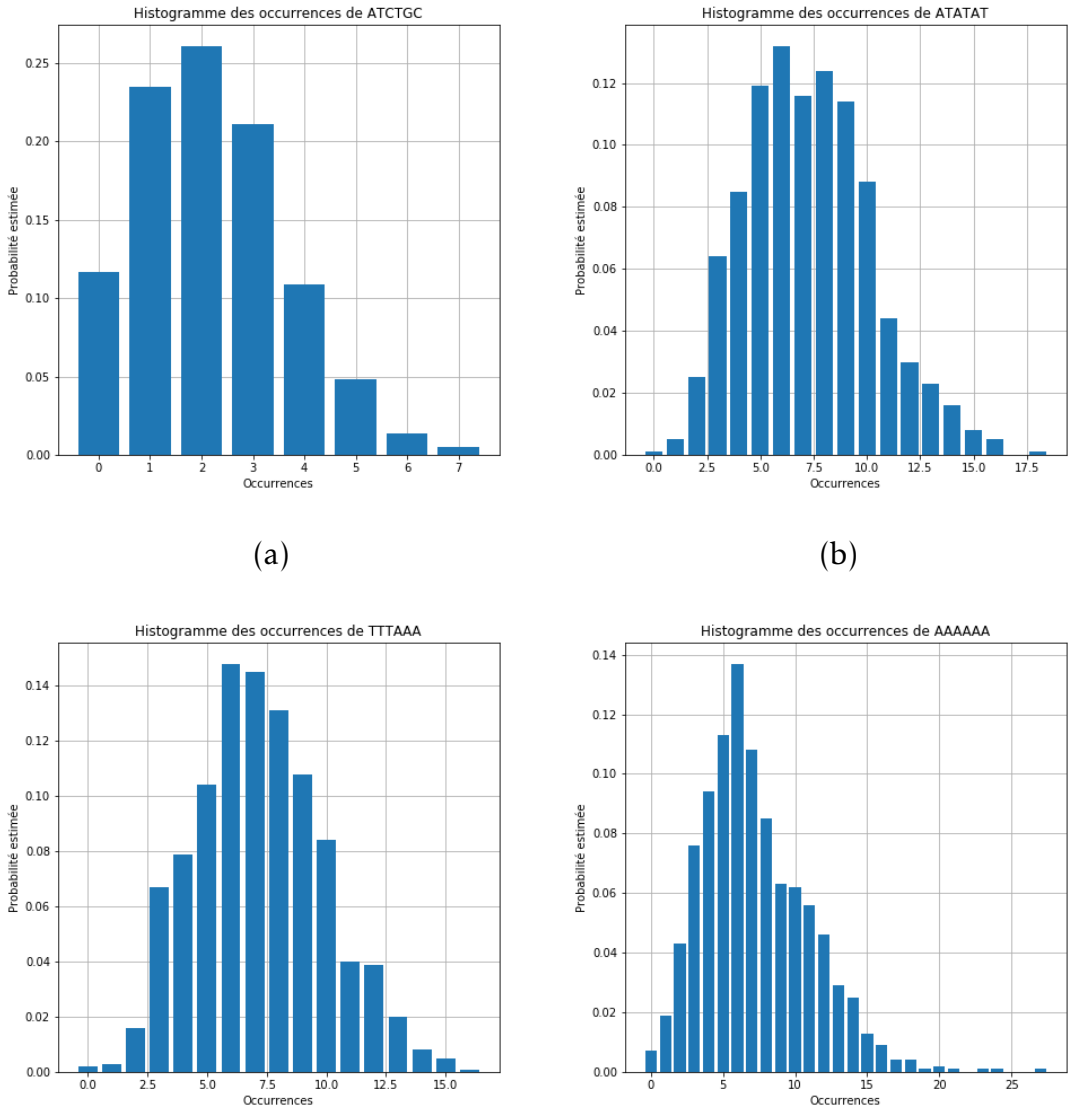

FIGURE 3 – Histogrammes avec les probabilités estimées du nombre d'occurrences des mots (a) ATCTGC, (b) ATATAT, (c) TTTAAA et (d) AAAAAA.

l'intervalle de confiance pour une loi de Bernoulli, donné par

$$I = \left[\hat{p} - t_\alpha \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}, \hat{p} + t_\alpha \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}\right]$$

où $\hat{p}$ est la valeur estimée de p et t_α est calculé à partir du niveau de confiance α, avec $t_\alpha \approx 1{,}96$ si $\alpha = 0{,}05$. La Figure 4 représente l'histogramme du nombre d'occurrences du mot ATATAT avec des barres représentant les intervalles de confiance de chaque probabilité calculés à l'aide de la formule ci-dessus et $\alpha = 0{,}05$. Les histogrammes avec les intervalles de confiance pour les autres mots sont donnés dans le fichier `Projet3.ipynb`.

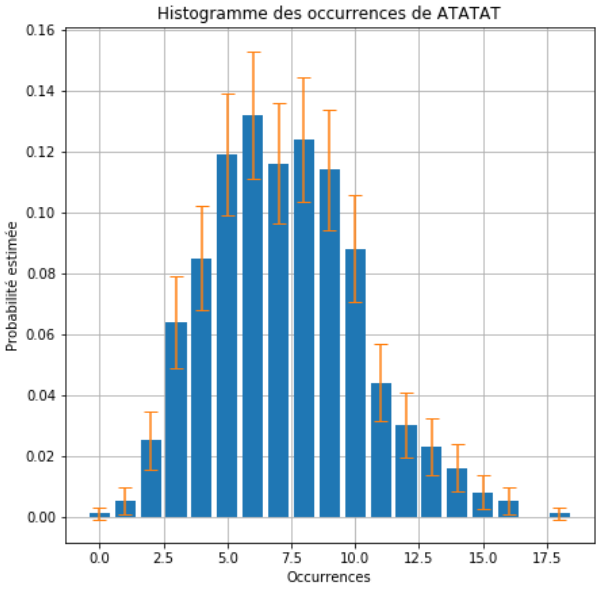

FIGURE 4 – Histogrammes avec les probabilités estimées et les intervalles de confiance pour le nombre d'occurrences du mot ATATAT.

3.3 Modèles de dinucléotides et trinucléotides

Question 1

Il s'agit d'une chaîne de Markov d'ordre 1 car seul le nucléotide précédent est utilisé pour déterminer la loi du prochain nucléotide de la séquence. Comme on s'attend à ce que tous les coefficients de la matrice M soient strictement positifs (car il n'y a pas de paire de nucléotides qui ne peuvent pas se suivre dans une séquence), la chaîne de Markov correspondant est ergodique. Alors la probabilité stationnaire peut être estimée par la fréquence des lettres dans une séquence obtenue par l'utilisation de la chaîne de Markov. En supposant que la séquence du fichier `yeast_s_cerevisae_genomic_chr1-4.fna` provient d'une chaîne de Markov, la probabilité stationnaire peut être estimée alors par la table de nucléotides obtenue à partir de l'application de la fonction `nucleotide_frequency` aux données de ce fichier.

Questions 2 et 3

Les fonctions `estim_M` et `simule_Markov` pour estimer la matrice M et pour simuler une séquence produite par la chaîne de Markov correspondante ont été codées dans `projet.py` et testées dans `Projet3.ipynb`. La matrice M estimée à

partir des données du fichier `yeast_s_cerevisae_genomic_chr1-4.fna` est

$$M = \begin{pmatrix} 0.32505 & 0.18552 & 0.20059 & 0.28884 \\ 0.33571 & 0.20925 & 0.15756 & 0.29747 \\ 0.31321 & 0.20245 & 0.20774 & 0.27659 \\ 0.24149 & 0.21034 & 0.22275 & 0.32541 \end{pmatrix}.$$

On observe en particulier que le modèle précédent où toutes les lettres de la séquence étaient considérées comme indépendantes n'est pas très adapté, puisque les lignes de la matrice M ne sont pas toutes identiques. On vérifie aussi, conformément à la question précédente, que le vecteur

$$m = \begin{pmatrix} 0.29984 & 0.20110 & 0.20000 & 0.29907 \end{pmatrix}$$

obtenu dans la Question 2 de la Partie 2 est bien une probabilité invariante pour M car $mM = m$.

Question 4

Soit le mot $w = w_0 w_1 \cdots w_{k-1}$ de longueur k et une séquence $S_0 S_1 S_2 \cdots$ obtenue à partir d'une chaîne de Markov de matrice de transition M et probabilité invariante m. La probabilité pour que w apparaisse à l'indice i de cette séquence est:

$$\begin{aligned} & P(S_i = w_0 \cap S_{i+1} = w_1 \cap \cdots \cap S_{i+k-1} = w_{k-1}) \\ &= P(S_i = w_0) P(S_{i+1} = w_1 | S_i = w_0) \cdots P(S_{i+k-1} = w_{k-1} | S_{i+k-2} = w_{k-2}) \\ &= m(w_0) M(w_0, w_1) \cdots M(w_{k-2}, w_{k-1}). \end{aligned}$$

Le calcul de cette probabilité a été fait dans la fonction `logproba_mot_Markov`, qui renvoie le logarithme de cette probabilité car elle peut parfois être très petite.

Question 5

Les mêmes arguments qu'à la Question 2 de la Partie 3.1 montrent que, en fixant un mot w de longueur k et une taille ℓ de séquence, la variable aléatoire X donnant le nombre d'occurrences de ce mot dans une séquence de taille ℓ satisfait

$$\mathbb{E}(X) = p(\ell - k + 1),$$

la seule différence étant que désormais on a, d'après la question précédente,

$$p = m(w_0) M(w_0, w_1) \cdots M(w_{k-2}, w_{k-1}).$$

On a ainsi codé une fonction `comptage_attendu_Markov` qui est très similaire à `comptage_attendu`, mais remplace l'appel à `logproba` par un appel à `logproba_mot_Markov`.

Question 6

On a comparé les nombre d'occurrences attendus dans le modèle de nucléotides où toutes les lettres sont indépendantes et dans le modèle de dinucléotides par chaîne de Markov d'ordre 1. Les résultats obtenus pour des mots de longueur 2 sur des séquences de taille 100 sont présentés dans le tableau ci-dessous :

Mot	Nucléotide	Dinucléotide
AA	8,9002	9,6486
AC	5,9695	5,5069
AG	5,9367	5,9543
AT	8,8774	8,5740
CA	5,9695	6,6836
CC	4,0038	4,1661
CG	3,9818	3,1369
CT	5,9541	5,9224
GA	5,9367	6,2015
GC	3,9818	4,0084
GG	3,9599	4,1133
GT	5,9214	5,4765
TA	8,8774	7,1500
TC	5,9541	6,2277
TG	5,9214	6,5952
TT	8,8546	9,6346

On remarque ainsi qu'il y a des différences : certains mots, comme AA, CA, GA ou TT, sont plus fréquents ; d'autres, comme AT, CG, GT ou TA, sont moins fréquents, et d'autres, comme AG, CT ou GC, ont eu leur fréquence quasiment inchangée. Les différences sont en général assez petites, car, même si la matrice M n'a pas toutes ses lignes identiques, elles sont néanmoins assez similaires au vecteur m contenant la probabilité invariante, ce qui indique que ce modèle de dinucléotide n'est pas très loin du modèle de nucléotide.

Question 7

On a recalculé les graphes représentant les nombres d'occurrences observés et attendus par le modèle de chaîne de Markov, qui sont représentés sur la Figure 5. On observe des comportements similaires à ceux de la Figure 1.

3.4 Probabilités de mots

Question 1

Comme à la Question 2 de la Partie 3.1, on fixe un mot $w = w_0 w_1 \cdots w_{k-1}$ de longueur k et on considère des séquences aléatoires S de longueur ℓ. On note par X_i la variable aléatoire Bernoulli qui vaut 1 si et seulement si le mot w apparait à la position i de S. Le nombre d'occurrences de w dans S est donc $X = \sum_{i=0}^{\ell-k} X_i$. En faisant l'hypothèse, comme indiqué à l'énoncé, que les différentes positions

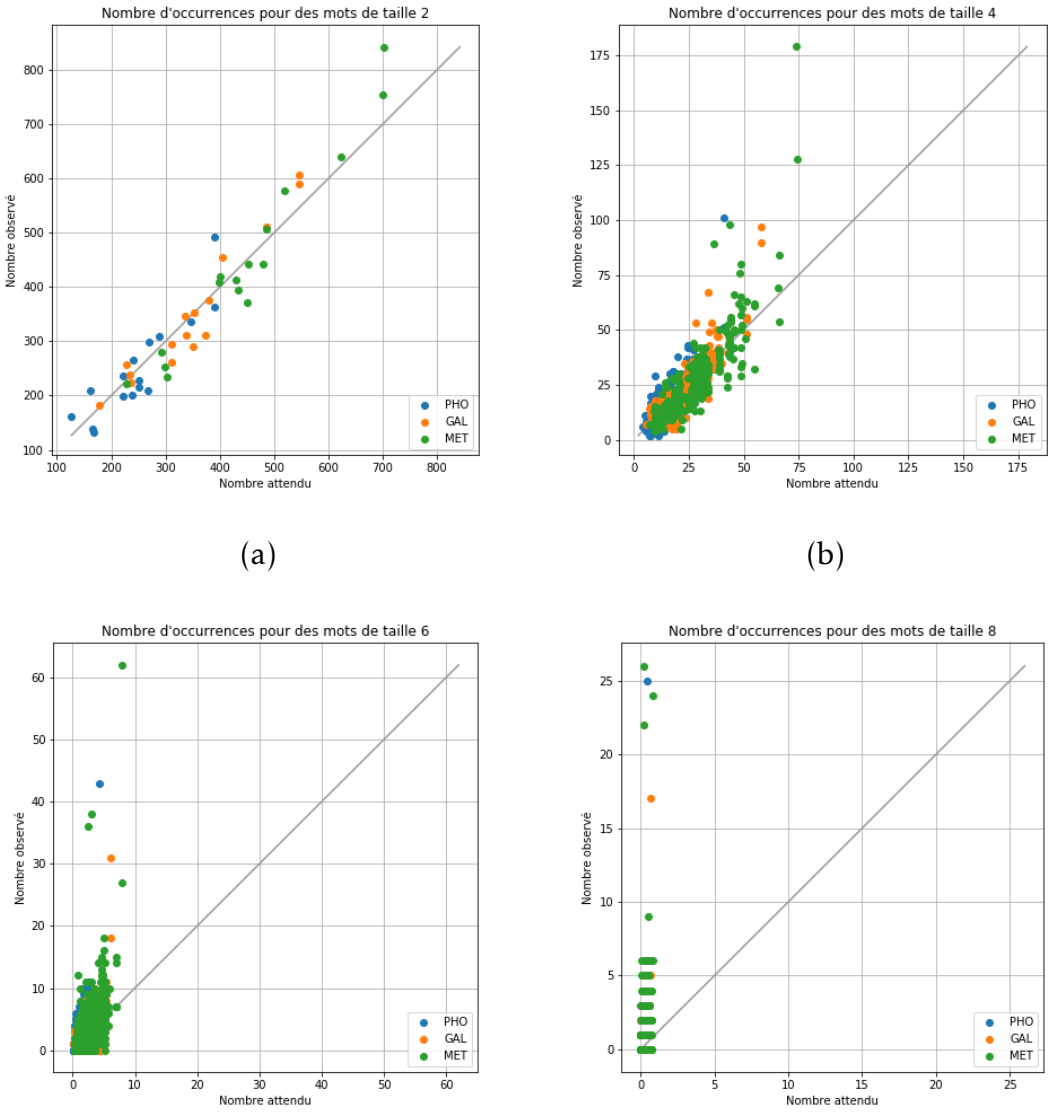

FIGURE 5 – Nombre d'occurrences attendu dans le modèle de chaîne de Markov et observé pour (a) $k = 2$, (b) $k = 4$, (c) $k = 6$ et (d) $k = 8$.

d'occurrences du mot w sont indépendantes, X est alors une somme de variables aléatoires indépendantes et Bernoulli de même paramètre p, et donc X suit une loi binomiale de paramètres p et $\ell - k + 1$. Rappelons que, d'après la Question 4 de la Partie 3.3, on a $p = m(w_0)M(w_0, w_1)\cdots M(w_{k-2}, w_{k-1})$.

Question 2

Si $\ell - k + 1$ est grand, on peut approcher la loi binomiale de X par une loi de Poisson de paramètre $\lambda = (\ell - k + 1)p$.

Question 3

On a calculé la distribution de probabilité empirique des mots ATCTGC, ATATAT, TTTAAA et AAAAAA en simulant 1000 séquences aléatoires de taille 10000 avec le modèle de dinucléotide par chaîne de Markov et tracé les histogrammes correspondants, qui sont données dans la Figure 6. On y représente aussi des points donnant la loi de Poisson avec le paramètre de la Question 2. On remarque que les histogrammes des probabilités empiriques des mots ATCTGC et TTTAAA, qui ne se chevauchent pas, sont très proches de la loi de Poisson de la Question 2. Même avec les chevauchements possibles, l'histogramme du mot ATATAT est lui aussi assez proche de la loi de Poisson correspondante. Cependant, à cause du grand nombre de chevauchements possibles, l'histogramme du mot AAAAAA s'écarte beaucoup de la loi de Poisson, indiquant que l'hypothèse d'indépendance utilisée à la Question 1 pour modéliser le nombre d'occurrences par une variable binomiale n'est pas pertinente dans ce cas.

Question 4

La fonction `proba_Poisson` calcule la probabilité d'occurrence d'un mot selon la loi de Poisson de la Question 2. Elle utilise pour cela le mot, la matrice de transition M de la chaîne de Markov et sa probabilité invariante m du modèle dinucléotide pour estimer la probabilité d'occurrence de ce mot dans une suite dont la longueur est passée en argument. Comme la loi de Poisson a un support infini, on s'arrête dès que la probabilité cumulée est supérieure à $1 - \varepsilon$, où ε est une valeur de tolérance passée en argument.

La fonction `proba_Poisson_geq_n` calcule, pour un mot et une longueur de séquence données, la probabilité $P(N \geq n)$, où N est une variable aléatoire de loi Poisson avec le paramètre donné à la Question 2. Elle utilise pour cela la fonction de répartition de la loi de Poisson déjà disponible dans le module `scipy.stats` de Python. Cette fonction est utilisée ensuite pour coder la fonction `proba_Poisson_from_list` qui, étant donnée une longueur de mot k, une séquence de nucléotides et une valeur de ε, détermine tous les mots w à k lettres tels que la probabilité que le nombre d'occurrences N_w de w dans une séquence aléatoire obtenue par une chaîne de Markov soit supérieur ou égal au nombre d'occurrences observé n_w dans la séquence est inférieure à ε, i.e., tous les mots w tels que $P(N_w \geq n_w) < \varepsilon$. Elle renvoie la liste de ces mots.

Question 5

On a appliqué la fonction `proba_Poisson_geq_n` de la question précédente aux séquences PHO, GAL et MET. Les résultats sont donnés dans le fichier `Projet3.ipynb`. On observe que, même pour des valeurs assez petites de ε, on a une grande quantité de mots qui apparaissent un nombre significativement plus qu'attendu dans ces séquences. Par exemple, pour la séquence PHO, on a les mots suivants avec $\varepsilon = 0{,}001$:

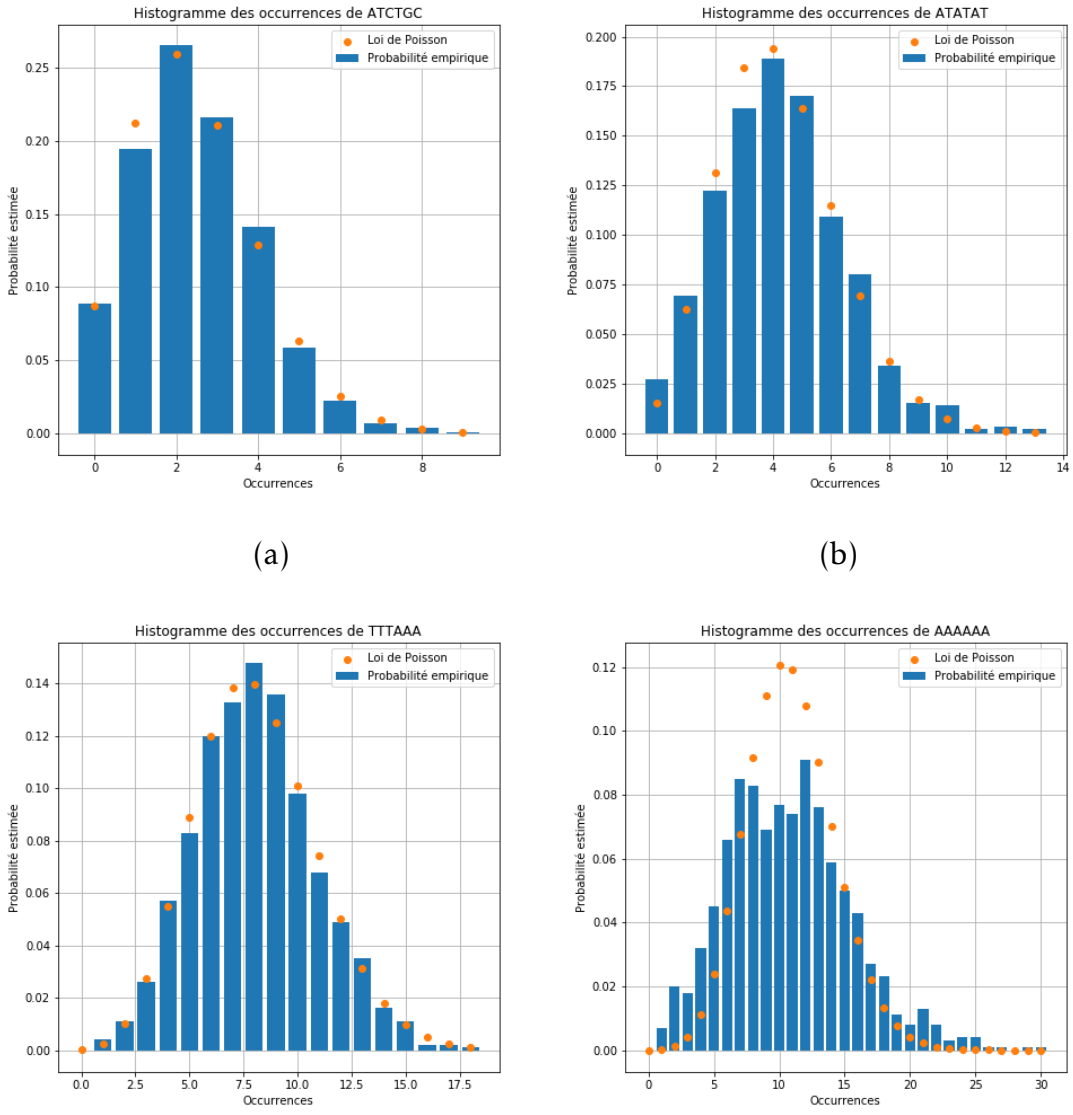

FIGURE 6 – Histogrammes avec les probabilités estimées du nombre d'occurrences et la loi de Poisson correspondante pour les mots (a) ATCTGC, (b) ATATAT, (c) TTTAAA et (d) AAAAAA.

— Mots de longueur 2 :
`AA GC`
— Mots de longueur 4 :
`AAAA AAGA ACGT AGAA CAGC CGTG TATA`
— Mots de longueur 6 :
`AAAAAA AAAAAG AAACGT AAATTA AAGAAA AAGAAG ACGTAT ACGTGC ACGTGG AGCAGC CAACAA CAAGAA CACGTG CGTGGG GAAGAA GCACGT GCAGCA GCAGCG TATATA TCATCT TGCCAA`
— Mots de longueur 8 :

AAAAAAAA AAAAAAAG AAAAAAGA AAAGTAAA AACAACAA AAGAAATG AAGAAGAA
ACACGTGG ACAGCAGC ACGCTCTC ACGTATAT ACGTGCAG AGAAGAAG AGCAGCGG
AGGAAGAA ATAGGCGC ATATAAGC CAAAAAAA CAAGAATG CACACGTG CACGTGGG
CAGCAGCG CCTCATCT CGGAAGAG CGTATATA CGTGCAGC CGTTTTCG CTCGTAAG
GAAAAATT GAAGAAGA GCGTATTA GGAAGAGC GTATATAA TAAAAAAA TACGGGAC
TATATAAG TCACACGT TCGTAAGG