Probabilités et statistiques

Analyse de séquences génomiques

Becirspahic Lucas
Vu Huc

Université Pierre et Marie Curie - 17 mars 2017

Introduction

Dans ce projet, nous analysons des séquences d'ADN pour chercher à comprendre le rôle des différents éléments du génome. Nous utilisons ainsi des méthodes statistiques, pour en particulier déterminer les mots qui correspondent à des sites de fixation des séquences promoteurs. Le langage Python est utilisé pour construire les fonctions et les graphiques qui permettent d'analyser la constitution d'une séquence.

1 Implémentation en Python

On dispose de quatre séquences génomiques : *S. cerevisae, PHO, GAL* et *MET*, qui contiennent respectivement 2 928 106, 4071, 5754 et 7323 nucléotides. Pour les manipuler plus facilement, on transforme chaque lettre en un entier compris entre 0 et 3 (ou -1 si elle n'appartient pas à $\{A, C, G, T\}$) qu'on range dans une liste. On possède également un certain nombre de fonctions de base : transformation d'un mot en lettres en une liste d'entiers, comptage des occurrences de lettres, calcul de la fréquence des lettres, simulation de séquence...

2 Description empirique

2.1 Redéfinition d'un mot

Il s'agit dans un premier temps de simplement dénombrer tous les mots de longueur k dans une séquence. On peut d'ores et déjà donner le nombre total de mots, qui est égal à 4^k : pour les k lettres du mot, on a pour chacune des lettres une valeur possible parmi $\{A, C, G, T\}$. Ensuite, il est nécessaire d'associer chaque un mot à un entier pour simplifier l'analyse, en l'écrivant en base 4. On définit alors un mot de la manière suivante :

$$\texttt{"TACG"} \rightarrow \texttt{[3, 0, 1, 2]} \rightarrow 3 \times 4^3 + 0 \times 4^2 + 1 \times 4^1 + 2 \times 4^0 \rightarrow 198$$

Naturellement, on dispose d'une fonction `invCode()` qui permet d'effectuer la manipulation inverse en utilisant le principe de la division euclidienne en base 4.

2.2 Comptage de mots expérimental

Supposons qu'on ait calculé la fréquence des lettres dans une séquence donnée. La fréquence d'apparition du mot dans une séquence de longueur l est égale au produit des fréquences de chacune des lettres le composant, et par conséquent, en multipliant par la longueur totale de la séquence, on obtient son nombre d'occurrences. On définit ainsi une fonction `comptage_attendu()`, qui renvoie un dictionnaire contenant le nombre d'apparitions attendues pour chaque mot.

```
1  def comptage_attendu(k,mots,freq,nb):
2      ''' k : longueur du mot, mots : liste de mots possibles (tableau d'entiers)
3      freq : tuple contenant la frequence attendue de chaque lettre,
4      nb : nombre total de nucleotides dans la sequence '''
5      dico = dict((key, 0) for key in mots)
6      for mot in mots:
7          proba = 1
8          tab_mot = invCode(mot, k)
9          for chiffre in tab_mot:
10             proba *= freq[chiffre]
11         dico[mot] = proba*nb
12     return dico
```

On définit également une fonction pour compter le nombre effectif d'occurrences de chaque mot de taille k dans la séquence, en comptant simplement parmi les $(l-k+1)$ mots. On compare ainsi le nombre d'occurrences observé d'un mot, dans un très grand échantillon de séquences, à son nombre d'occurrence attendu, obtenu en connaissant la fréquence des lettres. Tous les graphiques obtenus, pour les séquences PHO, GAL et MET, et $k \in \{2, 4, 6, 8\}$, sont référencés dans l'annexe A.

On remarque qu'il existe toujours au moins un mot dont le nombre d'occurrences observé est supérieur au nombre attendu. En particulier, pour PHO et $k = 6$, il n'existe qu'un seul mot qui apparait significativement plus souvent que prévu, alors qu'il y en a deux pour GAL, et un plus grand nombre encore pour MET. Ces motifs pourraient donc correspondre à des sites de fixation des facteurs de transcription. Pour s'en assurer, on décide d'effectuer un grand nombre de simulations pour évaluer la validité de ces résultats.

3 Simulation de séquences aléatoires

On simule ici 1000 séquences aléatoires pour tenter d'estimer la probabilité d'observer un mot plus de n fois dans une séquence de longueur l. Il suffit simplement de compter le nombre de séquences de la simulation où un mot w a été effectivement observé plus de n fois, qu'on note par la variable aléatoire N. On obtient alors une valeur pour la probabilité empirique qu'on note $P_{emp}(N \geq n)$.

Une démarche intéressante est de calculer P_{emp} pour des mots qui ont une probabilité d'apparition identique (ou au contraire, dont tous les caractères sont distincts). D'après les histogrammes obtenus en figure 1 et 2, par exemple, le mot `AAAAAA` a une plus grande probabilité d'apparaître que le mot `ATCTGC`. Les mots `ATATAT` et `TTTAAA` sont composés des mêmes lettres mais dans un ordre différent, et leur probabilité d'apparition est, comme prévu, quasi identique. Si la fréquence d'apparition de chaque lettre était égale, alors le mot `AAAAAA` aurait moins de chances d'apparaître que le mot `ATCTGC`, qui est constitué de lettres distinctes.

On se propose de calculer un intervalle de confiance pour cette probabilité expérimentale. Soit p la probabilité réelle que le mot apparaisse au moins n fois. Posons $p' = P_{emp}(N \geq n)$ et $1 - p' = P_{emp}(N < n)$. N suit une loi binomiale. Comme on a un grand nombre, s, de simulations, on peut approximer N par une loi normale centrée en 0 grâce au théorème central limite. On définit alors l'intervalle de confiance à 95% (σ étant l'écart-type réel) :

$$I = \left[p - t_\alpha \frac{\sigma}{\sqrt{s}}, p + t_\alpha \frac{\sigma}{\sqrt{s}} \right]$$

Le problème qui se pose est qu'on ne connaît pas cet écart-type réel. Deux manières de résoudre le problème s'offrent à nous :

- On fait une approximation de l'écart-type réel à partir de l'écart-type expérimental
- D'après la loi des grands nombres, l'écart-type expérimental va converger vers l'écart-type réel

Dans notre cas, on décide d'utiliser l'écart-type expérimental, et on obtient donc l'intervalle de confiance à 95% suivant :

$$I = \left[p' - 1.96 \frac{\sigma_{emp}}{\sqrt{s}}, p' + 1.96 \frac{\sigma_{emp}}{\sqrt{s}} \right]$$

Dès lors, les résultats empiriques ne suffisent pas, et on cherche à vérifier la cohérence de nos résultats en calculant les probabilités théoriques d'apparition des mots.

4 Probabilités de mots

4.1 Calcul théorique

Pour pouvoir estimer la cohérence de notre probabilité empirique, calculons une approximation de la probabilité qu'un mot w apparaisse n fois dans une séquence aléatoire à l'aide de lois usuelles.

Soit P_{th} la probabilité d'avoir un mot w, de longueur k, qui apparaît n fois dans une séquence aléatoire de longueur l, avec $0 \leq n \leq (l-k+1)$. Soit N la variable aléatoire qui compte le nombre de fois où le mot apparaît dans une séquence. On suppose sur les positions d'occurrences d'un mot sont indépendantes. Notons qu'on peut calculer P_{th} par la formule suivante, quelle que soit l'approche abordée :

$$P_{th}(N \geq n) = 1 - P_{th}(N < n) = 1 - \sum_{i=0}^{n-1} P_{th}(N = i)$$

– Première approche
Pour les $(l-k+1)$ positions différentes de la séquence (les mots ne se chevauchent pas), il y a un seul choix de mot sur tous les mots possibles. Or, il existe 4 bases nucléiques possibles (`A`, `C`, `G` ou `T`) pour chaque lettre du mot de longueur k, il y a donc 4^k mots possibles. Ainsi, N suit une loi binomiale :

$$N \sim \mathcal{B}(l-k+1, \frac{1}{4^k})$$

Posons $q = l-k+1$. On obtient alors :

$$P_{th}(N = n) = C_q^n \left(\frac{1}{4^k}\right)^n \left(1 - \frac{1}{4^k}\right)^{q-n}$$

On se dote d'une fonction simple, qui calcule cette probabilité selon la formule ci-dessus.

Calculons la probabilité théorique qu'un mot de longueur 6 (par exemple `ATCTGC`) soit présent au moins une fois dans la séquence PHO. On trouve ainsi $P_{th}(N \geq 1) = 1 - P_{th}(N = 0) \approx 0.629$. Or, la valeur empirique de cette probabilité est 0.544, une différence relativement grande pour considérer le résultat théorique trouvé incohérent. En effet, dans cette approche, on ne prend pas en compte les fréquences de chaque lettre, au contraire de la probabilité empirique. De plus, si on regarde les probabilités pour les quatre mots donnés dans l'énoncé (`ATCTGC`, `ATATAT`, `TTTAAA`, `AAAAAA`), la valeur est identique pour les quatre mots alors qu'on trouve des probabilités empiriques bien différentes. Cette approche est donc fausse, il faut aborder le problème autrement.

– Seconde approche
Soit $w = w_1 w_2 ... w_k$ le mot dont on veut déterminer le nombre d'apparitions. Soit $\mathcal{F}(w_i)$ la fréquence de la lettre w_i dans une séquence. La probabilité que w apparaisse est le produit des probabilités de chacune de ses lettres. La variable aléatoire N suit alors une loi binomiale dont on a modifié un paramètre :

$$N \sim \mathcal{B}(l-k+1, \prod_{i=1}^{k} \mathcal{F}(w_i))$$

Ainsi, en posant comme précédemment $q = l-k+1$, on obtient :

$$P_{th}(N = n) = C_q^n \left(\prod_{i=1}^{k} \mathcal{F}(w_i)\right)^n \left(1 - \prod_{i=1}^{k} \mathcal{F}(w_i)\right)^{q-n}$$

En calculant la probabilité que le mot ATCTGC apparaisse au moins une fois, étant donné le quadruplet des fréquences des lettres dans la séquence PHO, on trouve $P_{th}(N \geq 1) \approx 0.542$, pour une probabilité empirique de 0.544. Cette approche est donc plus précise que la précédente.

Le temps de calcul de C_q^n étant relativement élevé, on cherche à approximer $P_{th}(N = n)$. On peut ainsi utiliser une loi de Poisson, en posant comme paramètre $\lambda = q \times \prod_{i=1}^{k} \mathscr{F}(w_i) = (l-k+1)\prod_{i=1}^{k} \mathscr{F}(w_i)$, ce qui donne :

$$P_{th}(N = n) \approx e^{-\lambda} \frac{\lambda^n}{n!}$$

De manière plus concrète, λ est le nombre moyen d'apparitions du mot w dans la séquence parmi les $(l-k+1)$ positions possibles, connaissant la fréquence de chaque lettre de w au préalable.

4.2 Vérification expérimentale sur séquence

Pour les quatre mots étudiés, on trouve à titre d'exemple les valeurs suivantes dans la séquence PHO, qui sont cohérentes (sauf pour AAAAAA, cela pourrait s'expliquer par le chevauchement des lettres, *c.f.* la partie sur les dinucléotides) :

Séquence	AAAAAA	ATATAT	TTTAAA	ATCTGC
$P_{th}(N = 1)$	0.996	0.955	0.955	0.542
$P_{emp}(N = 1)$	0.975	0.949	0.961	0.544
$P_{th}(N = 3)$	0.913	0.600	0.600	0.045
$P_{emp}(N = 3)$	0.801	0.596	0.576	0.043

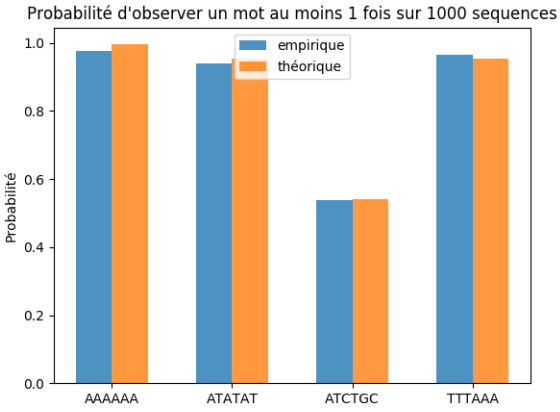

FIGURE 1 – $P(N \geq 1)$

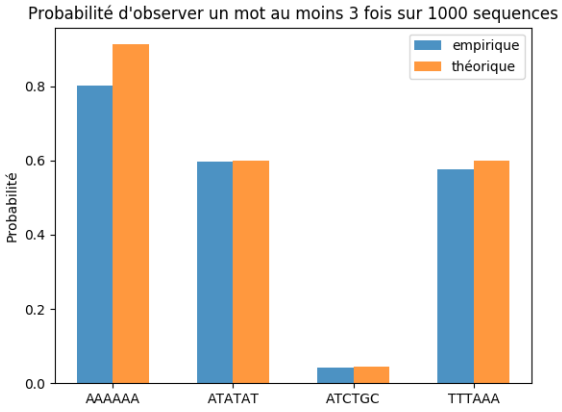

FIGURE 2 – $P(N \geq 3)$

On peut finalement déterminer, pour une longueur k de mot, les mots qui apparaissent significativement plus qu'attendu, en comparant la probabilité théorique et la probabilité empirique, i.e. si $P_{th}(N \geq n) < P_{emp}(N \geq n)$. On peut également fixer un facteur δ pour le sens du mot "significativement", ce qui conduirait à une condition $\delta \times P_{th}(N \geq n) < P_{emp}(N \geq n)$.

On arrive rapidement à un problème d'arrondi quand n et la longueur de la séquence deviennent grands. En effet, l'algorithme parvient à calculer la probabilité théorique, mais compte tenu des approximations du système, la probabilité empirique calculée est presque toujours égale à 0. On ne peut pas reconstruire les mots qui pourraient s'apparenter à des sites de fixation dans ces conditions, et par conséquent, il faut soit utiliser

les log-probabilités, soit définir un nouveau modèle pour calculer la probabilité expérimentale d'apparition de chaque mot.

5 Modèles de dinucléotides

On souhaite maintenant prendre en compte que certaines combinaisons de nucléotides ont une plus grande probabilité d'apparaître que d'autres, c'est-à-dire qu'il faut prendre en compte les chevauchements. Pour cela, on construit une matrice M à 4 lignes et 4 colonnes, où $M(i, j)$ est la probabilité de la lettre j sachant qu'on est sur la lettre i.

Pour construire la matrice M à partir des comptages de mots de longueur 2, pour chaque mot $w = (w_1, w_2)$, on incrémente $M(w_1, w_2)$ par le nombre d'occurrences de w dans la séquence. On obtient ainsi le nombre d'occurrences d'une lettre sachant la lettre précédente. Il suffit de sommer les valeurs sur une ligne (qui représentent toutes les lettres possibles) et de diviser chaque case par cette somme. Chaque ligne de la matrice indique donc la probabilité d'avoir une lettre sachant la lettre précédente.

	A	C	G	T
A				
C				
G				
T				

Une fois la matrice M construite, on souhaite l'utiliser pour calculer la probabilité d'apparition d'un mot à une position donnée. Soit $w = w_1 w_2 ... w_k$ un mot de longueur k. Par exemple, pour $k = 2$, on a :

$$P(w) = P(A)P(w_1|A)P(w_2|w_1) + P(C)P(w_1|C)P(w_2|w_1) + P(T)P(w_1|T)P(w_2|w_1) + P(G)P(w_1|G)P(w_2|w_1)$$

Ainsi, pour un mot de longueur k, son nombre attendu d'occurrences dans une séquence de taille l est égal à

$$(l - k + 1) \times P(w)$$

Il s'agit ensuite de comparer les deux modèles, nucléotides et dinucléotides, sur les comptages du nombre d'occurrences.

(Nous n'avons pas eu le temps de traiter les deux dernières question de la partie 3.4)

Conclusion

La détermination des sites de fixation dans une séquence génomique est un problème complexe. Après analyse des séquences, on peut en déduire qu'il existe bel et bien des mots dont la probabilité d'apparition observée est supérieure à la probabilité théorique. Cependant, il faut être capable de vérifier la cohérence et la validité des résultats empiriques, en choisissant le bon modèle à utiliser.

Annexe

A Comparaison nombre de mots attendu-observé

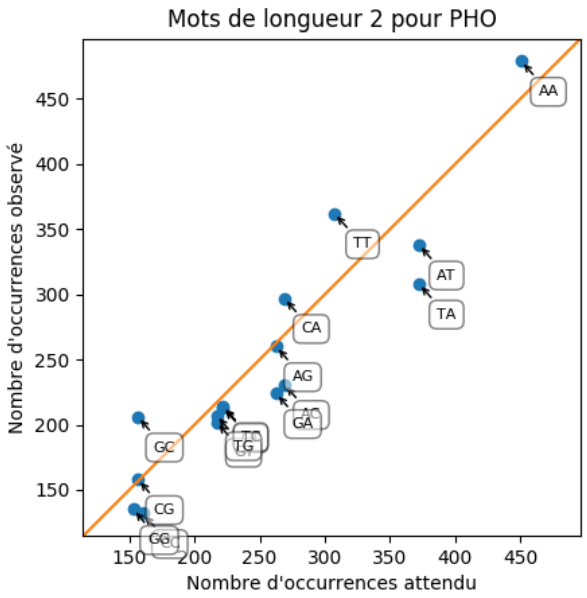

FIGURE 3 – PHO, k = 2

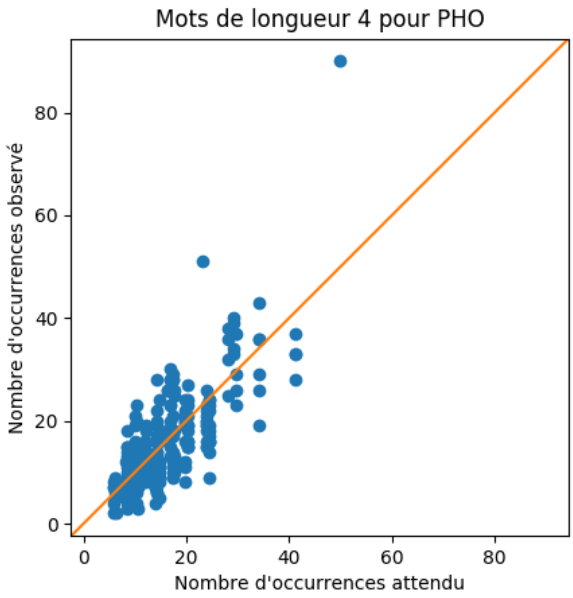

FIGURE 4 – PHO, k = 4

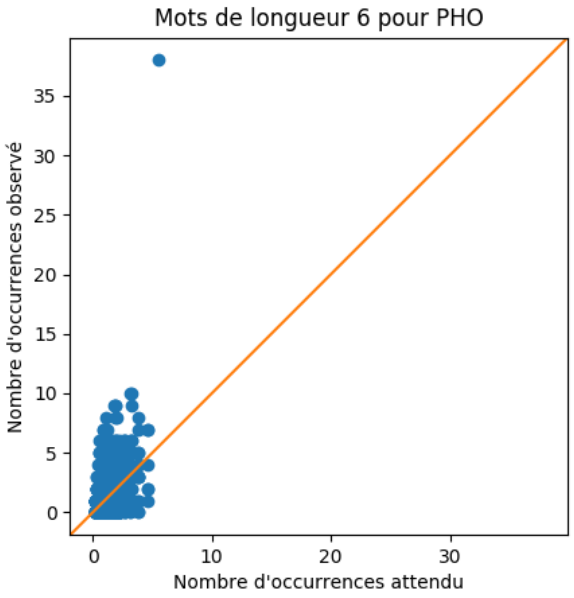

FIGURE 5 – PHO, k = 6

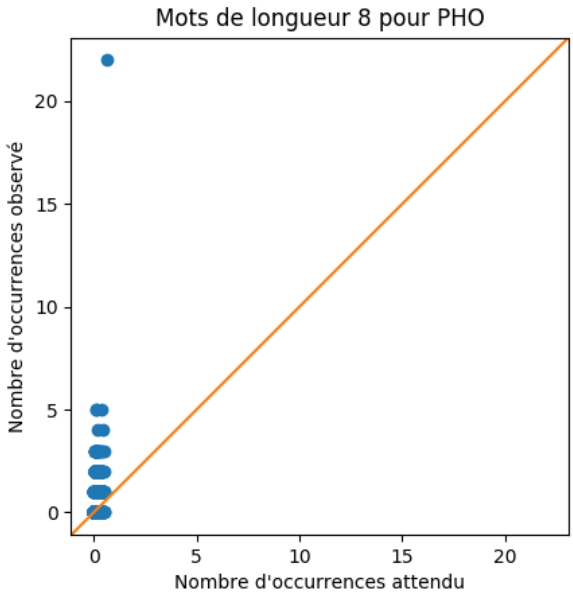

FIGURE 6 – PHO, k = 8

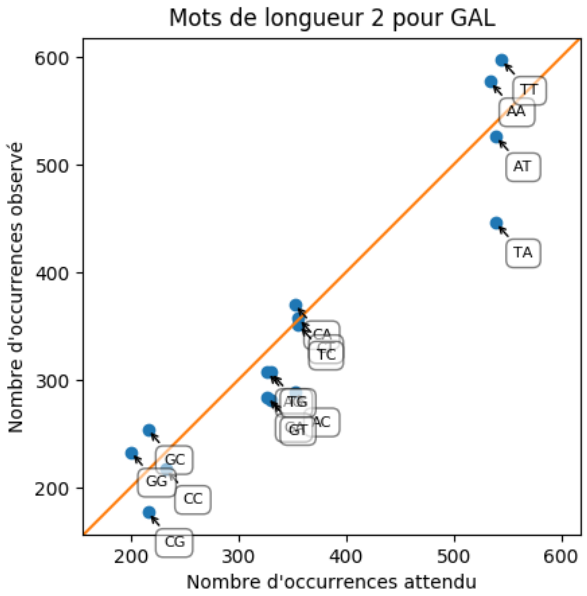

FIGURE 7 – GAL, k = 2

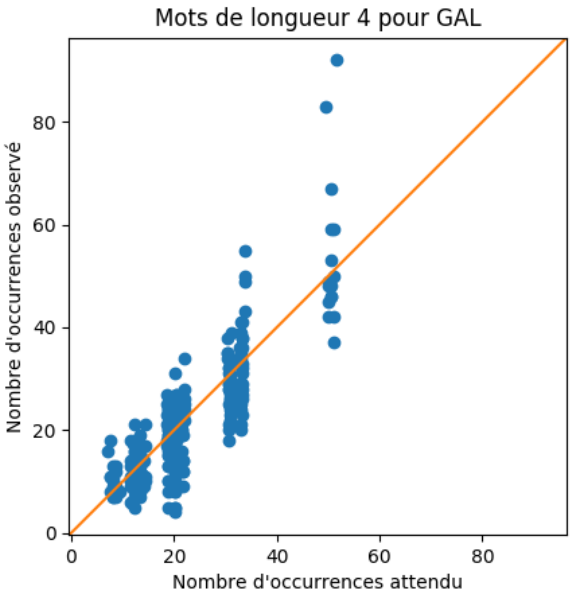

FIGURE 8 – GAL, k = 4

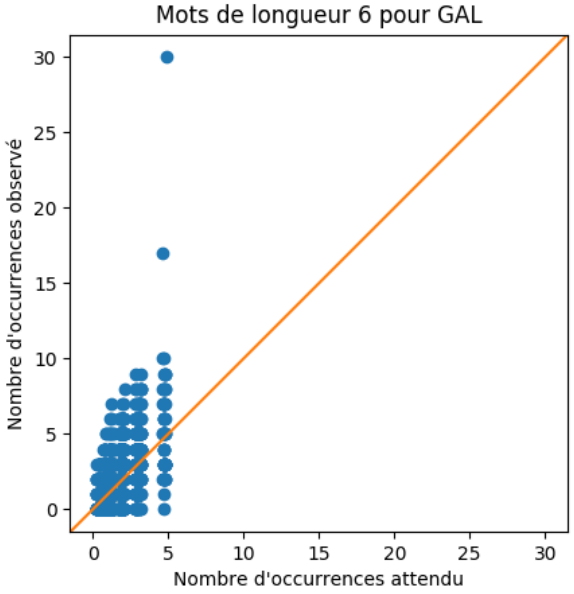

FIGURE 9 – GAL, k = 6

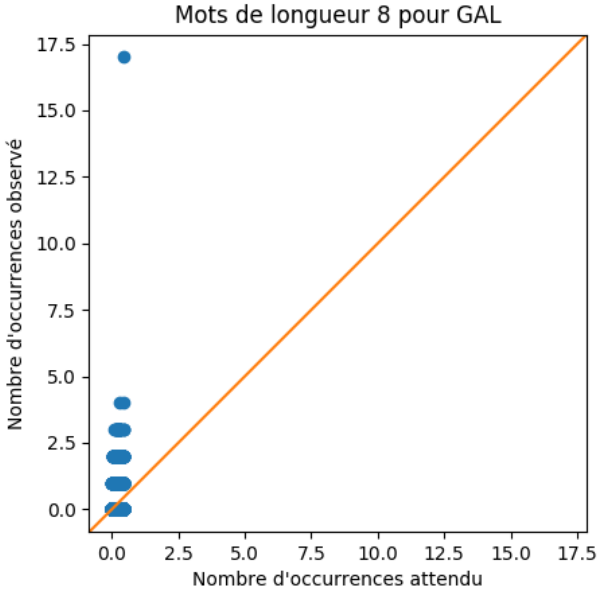

FIGURE 10 – GAL, k = 8

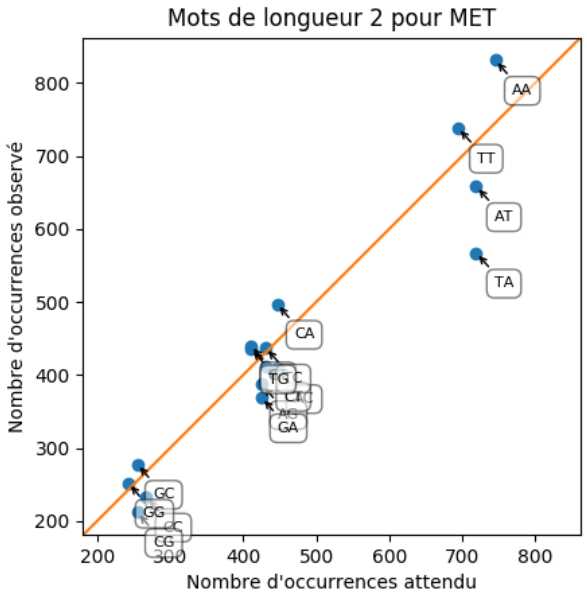

FIGURE 11 – MET, k = 2

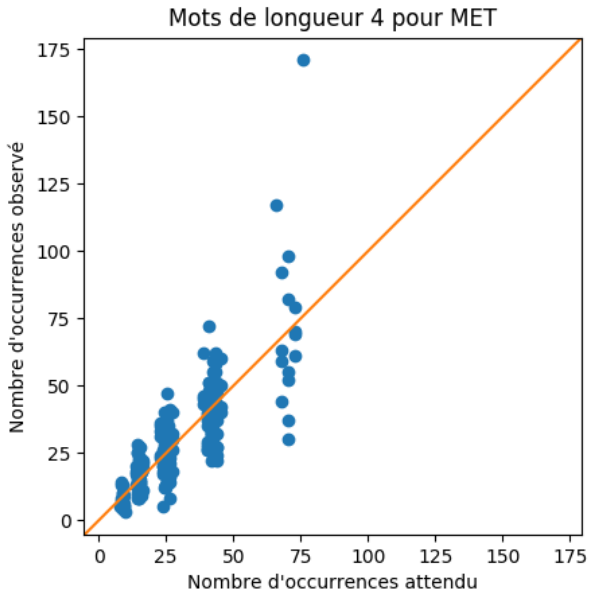

FIGURE 12 – MET, k = 4

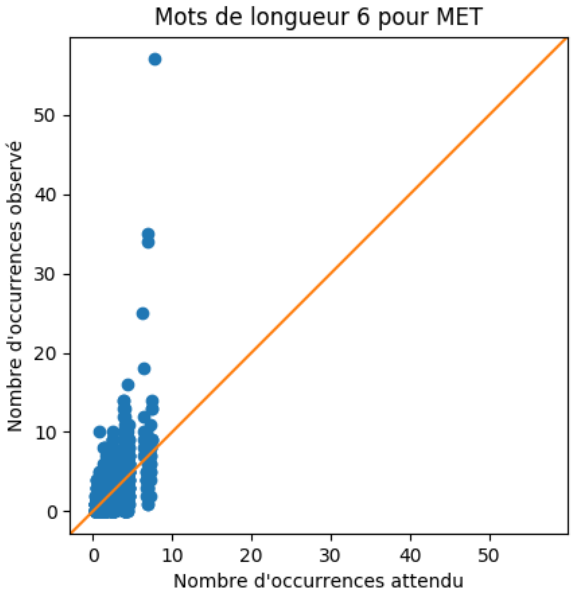

FIGURE 13 – MET, k = 6

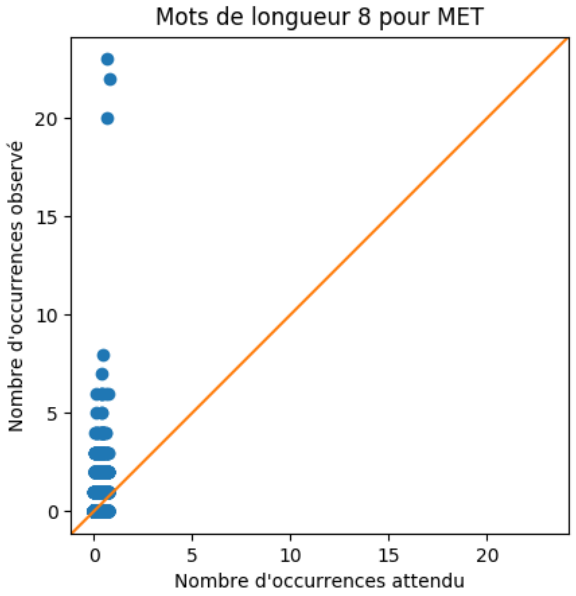

FIGURE 14 – MET, k = 8