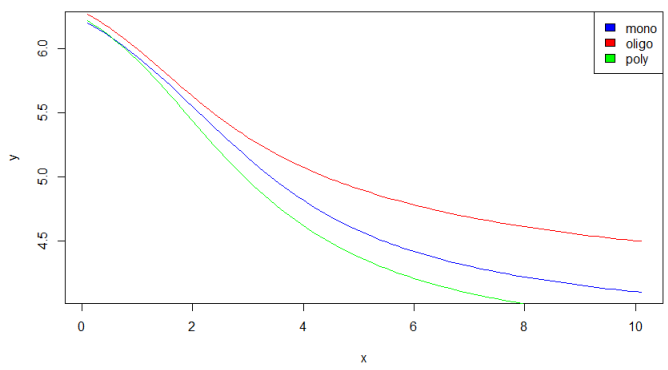
Implémentation des outils en Python

Indice de diversité de Hill

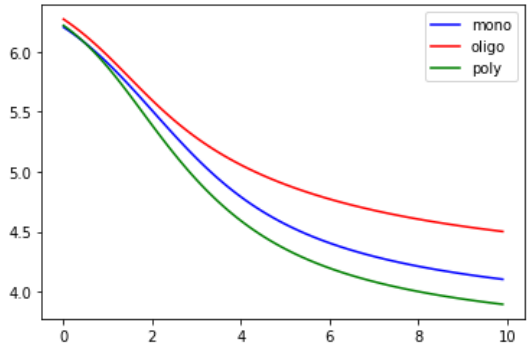
L'indice de diversité de Hill pour une population de N espèces, où p_i est la proportion de l'espèce i, se calcule avec la formule suivante :

$$ {}^{\alpha}H = \left(\sum_{i=1}^{N} p_i^{\alpha} \right)^{\frac{1}{1-\alpha}} $$

On se base sur le package vegan utilisé par l'outil SumRep, plus précisemment sur la fonction renyi(). On calcule cette valeur pour α compris entre 0 et 10 avec un pas de 0,1.
Pour obtenir les mêmes résultats avec l'implémentation en Python, nous calculons le logarithme de l'indice de diversité.



SumRep



Python

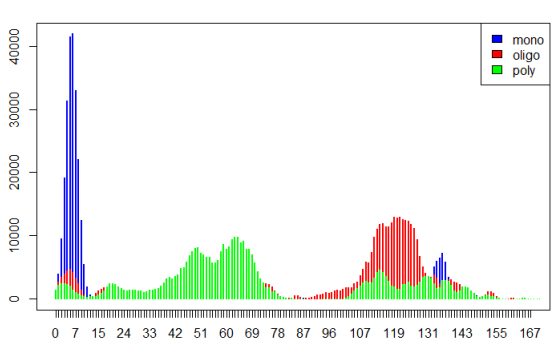
FIGURE 1 – Comparaison des résultats pour le calcul de l'indice de diversité de Hill entre SumRep et l'implémentation en Python.

Pairwise Distance Distribution

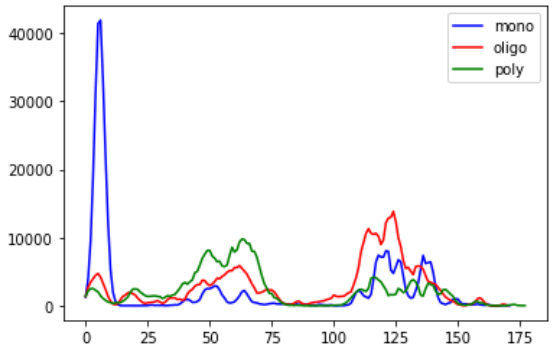
Le calcul de la Pairwise Distance Distribution effectué avec le package ImmuneRef utilise la distance de Levenshtein, qui correspond au nombre minimal de caractères à supprimer, insérer ou remplacer pour passer d'une chaîne de caractères à une autre.
Cette distance se calcule avec la formule de récurrence suivante :

$$ l(a,b) = \begin{cases} \max(|a|,|b|) & \text{si } \min(|a|,|b|) = 0, \\ \text{lev}(a-1,b-1) & \text{si } a[0] = b[0], \\ 1 + \min \begin{cases} \text{lev}(a-1,b) \\ \text{lev}(a,b-1) \\ \text{lev}(a-1,b-1) \end{cases} & \text{sinon.} \end{cases} $$

Pour l'implémentation en Python, nous avons donc créé une fonction permettant de calculer la distance de Levenshtein entre deux chaînes de caractères.
Il suffit ensuite de calculer cette distance pour chaque paire de séquence.
Cependant, l'utilisation de la fonction implémentée pour le calcul de toutes les distances implique un temps d'exécution trop grand. Nous avons donc utilisé le module Levenshtein en Python pour réduire le temps d'exécution.



SumRep



Python

FIGURE 2 – Comparaison des résultats entre SumRep et l'implémentation en Python.

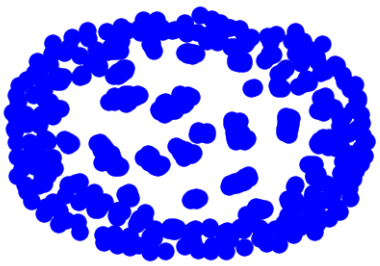
Réseaux

On peut tracer un réseau sous la forme d'un graphe dont les nœuds représentent les séquences et deux séquences sont liées lorsque leur distance de Levenshtein est égale à 1.
Il est alors intéréssant de compter le nombre de composantes connexes et le degré moyen des nœuds. Plus il y a de composantes connexes et plus le degré moyen est bas, plus il y a de diversité dans le répertoire.

En considérant les clones :

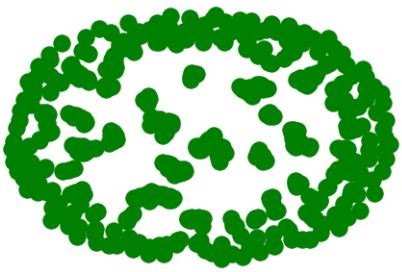
Number of connected components : 148
Average degree : 1.4412955465587045

Répertoire monoclonal



Number of connected components : 202
Average degree : 1.25

Répertoire oligoclonal



Number of connected components : 203
Average degree : 1.2055888223552895

Répertoire polyclonal

FIGURE 3 – Comparaison des graphes obtenus pour chaque répertoire avec Python.

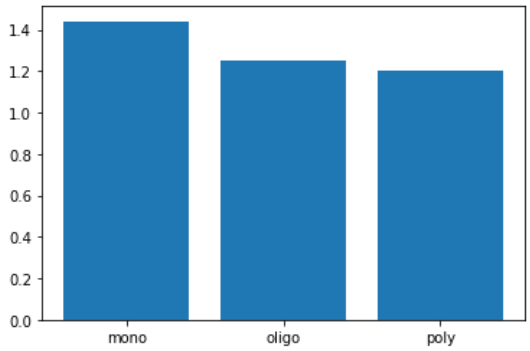
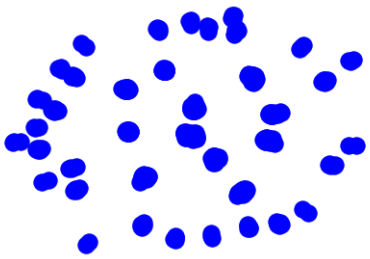


FIGURE 4 – Degré moyen des nœuds en fonction du répertoire avec Python.

En supprimant les clones :

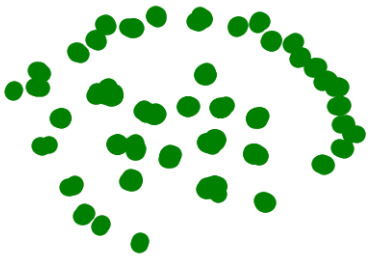
```
Number of connected components :  35
Average degree :  123.25101214574899
```

Répertoire monoclonal



```
Number of connected components :  40
Average degree :  22.276515151515152
```

Répertoire oligoclonal



```
Number of connected components :  42
Average degree :  15.2375249500998
```

Répertoire polyclonal

FIGURE 5 – Comparaison des graphes obtenus pour chaque répertoire avec Python.

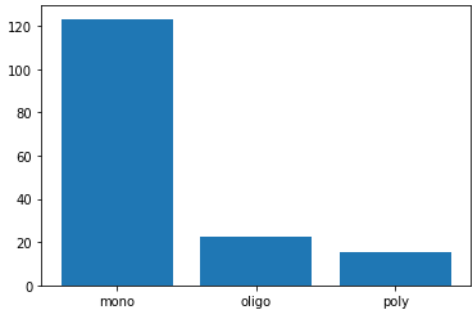


FIGURE 6 – Degré moyen des nœuds en fonction du répertoire avec Python.