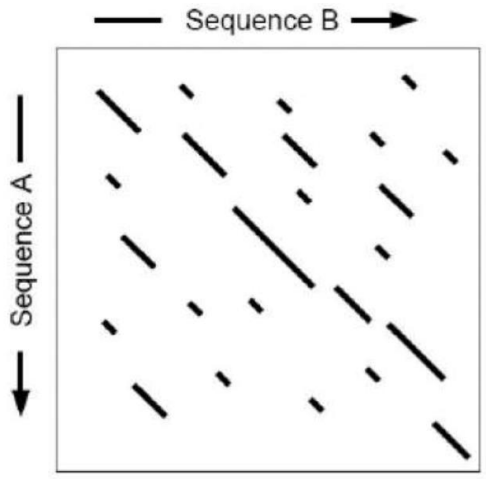
Dot Plot

Introduction

Un dot-plot est une technique permettant de comparer visuellement deux séquences (nucléiques ou protéiques), i.e., permettant de faire apparaître les segments (nucléiques ou protéiques) que "partagent" les deux séquences (soit strictement soit en utilisant un seuil de "ressemblance"). Ces segments similaires, on le verra sur l'exemple ci-après, sont spécifiés par des "diagonales" dans un espace à deux dimensions, chaque dimension étant associée à une des deux séquences. Ainsi, par exemple, faire un dot-plot des séquences nucléiques "TGCTATT" et "AAGCTAGG" permet de faire émerger une diagonale (cf. le schéma ci-dessous) spécifiant que le segment " GCTA " est commun aux deux séquences. L'idée du dot-plot est d'afficher un "X" au croisement d'une ligne et d'une colonne si la ligne et la colonne correspondent à la même lettre (i.e., si le ième caractère de la première séquence est égal au jème caractère de la seconde séquence) et un " " sinon.

```
    T   G   C   T   A   T   T
A                   X
A                   X
G       X
C           X
T   X           X       X   X
A                   X
G       X
G       X
```

Nous avons vu en cours "Comparaison de séquences" que le dot-plot est un outil, simple mais puissant, qu'il est pertinent d'employer lorsqu'on souhaite comparer deux séquences entre elles. On donne ci-après un exemple de dot-plot non trivial, où les diagonales représentent les segments des séquences A et B qui se ressemblent.



L'objectif de cette séance est de programmer un outil de ce style.

Pour afficher nos «dotplots», nous allons utiliser notamment des « fonctions graphiques ». Ces «fonctions» sont accessibles via une « librairie python » appelée «matplotlib.pyplot». Afin de prendre en main la librairie «matplotlib.pyplot», voici un exemple d'instructions permettant d'afficher 2 points dans un graphique :

```
import matplotlib.pyplot as plt
# importation de la bibliothèque nécessaire à la génération de graphiques avec l'acronyme plt
plt.title("Graphique") # titre du graphique
plt.xlabel("Axe X") # label/nom de l'axe des abscisses
plt.ylabel("Axe Y") # label/nom de l'axe des des ordonnees
plt.xlim(0, 100) # taille/longueur de l'axe des "x"
plt.ylim(0, 50) # taille/longueur de l'axe des "y"
plt.scatter(10, 20, c = "black", s=10)
# trace un point d'abscisse 10, d'ordonnee 20 (y = 20), de couleur noire (c = 'black'), et de taille 10 pixels (s = 10)
plt.scatter(50, 30, c = 'red', s = 20)# trace un point d'abscisse 50, d'ordonnee 30, de couleur rouge (c = 'red'), et de taille 20 pixels (s = 20)
plt.show() # affichage du graphique
```

ATTENTION : l'affichage en scatter des « plots » est lent…
Pour plus d'informations, vous pouvez consulter le site : http://www.python-simple.com/python-matplotlib/pyplot.php

Si on fait tourner un programme faisant une comparaison lettre à lettre avec les séquences «AAGCTAGG» et «GCTA» par exemple, on se rend compte que certains "points" viennent "parasiter" le résultat du dotplot, i.e., ces "point " ne font pas partie d'une diagonale (ils marquent simplement le fait que les séquences partagent de temps en temps des " lettres " identiques). Afin d'éliminer ce bruit, au lieu de comparer chaque lettre de la séquence en «X» avec chaque lettre de la séquence en «Y», on effectue cette comparaison par segment de plusieurs lettres ("fenêtre"). Enfin, une souplesse est nécessaire car si l'introduction d'une fenêtre permet bien de nous "débarrasser" des "X" ou points qui viennent parasiter/bruiter la lecture du "dot-plot" (et donc l'émergence des diagonales qui marquent le partage d'une similarité entre les deux séquences considérées), l'égalité stricte est trop stringente ce qui a tendance à "casser" (ou rompre) les diagonales.

Votre version du dot-plot devra donc dépendre d'un "seuil d'identité" (% d'identité minimale entre les 2 fenêtres) dont la valeur sera spécifiée par l'utilisateur ; si le pourcentage d'identité entre les deux "fragments (courants)" comparés est supérieur au seuil, un "X" ou 1 point sera affiché (rien dans le cas contraire).

On prendra par ailleurs soin de sauvegarder le dot-plot dans un fichier afin de doubler la sortie sur l'écran d'une sauvegarde pérenne.

Exercice 1 :

Réaliser un programme permettant d'effectuer un dot-plot à partir de 2 séquences fasta enregistrées dans des fichiers séparés. Le manipulateur donnera :

- Les noms des 2 fichiers contenant les séquences
- La taille de la fenêtre
- Le seuil d'identité minimal
- Le nom du fichier de sortie

Votre programme réalisera le dot-plot et sauvegardera l'image dans un fichier image (png) ainsi qu'une version du dot-plot en format texte. 2 fichiers (Dotseq1 et Dotseq2) vous sont fournis pour tester votre programme. Les légendes des axes doivent indiquer le numéro d'identification des séquences (ex . 133711986 pour la séquence du fichier Dotseq1).

Exercice 2 :

Récupérer les séquences (format "fasta") des deux protéines suivantes :
http://www.uniprot.org/uniprot/P0AA25.fasta
http://www.uniprot.org/uniprot/P14949.fasta
Faites tourner votre "dot-plot" en faisant varier la taille de la fenêtre et le seuil d'identité afin de faire apparaître les similarités pertinentes entre les deux séquences.

Vous enverrez votre programme (Nom_Prénom.py) ainsi que votre analyse de l'exercice 2 (interprétation du dotplot et justification des valeurs des paramètres choisis) au format odt/docx par mail.