TP: ADN Mining - Alignement

Vous trouverez le nécessaire ici : http://www.math-info.univ-paris5.fr/~lomn/Cours/BC/

Perl ou Python ? Ou autre

https://www.biostars.org/p/2737/ - 2011
« CPAN do your life more easy, and I think that is one of the missing pieces in Python. »– 2013

« There are also other languages to consider such as Ruby, with the advantage that is even newer than Python and it uses Gems, so it is easy to install new modules, or Java, perhaps more difficult to learn, but with much better performance than Perl, Python or Ruby. Perhaps we can get one idea of the use of these languages in the scientific community looking into number of citations of Bioperl (Stajich JE et al 2002), Biopython (Cock PJA et al 2009), Bioruby (Goto N. et al 2010) and Biojava (Prlic A et al. 2012) in 2012. Bioperl 128, Biopython 75, Bioruby 20 and Biojava 2. So I agree with Dr. Podicheti about Python is catching up Perl, but for now is one of the most used languages in bioinformatics. »

« Perl has a problem. It kind of encourages writing a short program which next month is faster to write again than understand what it's doing. One needs discipline to write readable code. Python, on the other hand, encourages readability. For me and my friends, ease of comprehending code written in Python was the reason to abandon Perl. Both own code and somebody's else. »

« Personally I am drifting away from Perl. I started out with Perl, I still own Perl for its support for me. But I find it is very difficult to transit from small programs to big programs in Perl. This can be done via OOP, but jumping from procedural Perl to OOP Perl was a pain for me. It may not be a problem to someone with good background in OOP, I started out with procedural, and Perl is not a good OOP starter. I can learn OOP with other languages, i.e. Python or Java and then come back to Perl? Not quite so, it is better learn Python and Java and keep using them if possible. »

« In addition, it is Big Data age, and to my experience, Perl is not doing much to it. Last time I tried with parallel and distributed Perl modules, they are very difficult to start up. I felt one has to be a hardcore Perl programmer and quite a geek in computer science to make use of them. At least that is what the authors of those modules are :-) On the other hand, it is much easier for me to understand and write parallel programs in Python or Java. The cause might be traced back to Perl's OOP and readability. »

« The fasta and genbank parsers of python are great for instance, but I think the phylo module in bioperl is better than the one in biopython (at least for the moment). »

« One place I find Perl useful is the command line, because of its stream processing and in-place editing capabilities. For me it is easier to use than awk and sed. To do things like taking the first, fourth and fifth column of a GFF file and writing them out in a chromosome:start-end format instead: $ cat myfile.gff | perl -p -i -e "substitution/regexp/here/"
Maybe Python and other languages have these possibilities too, but the terse code of Perl is not only more tolerable on the command line for me, it is outright desirable there. »

« I am a masters student and I have worked with both Perl and Python. Python was easy for me to learn than Perl. but when I learnt Perl i felt it was so useful in Bioinformatics to analyse sequences and others. But when i learnt python and started working with it I felt it is more easy to dynamically segment the data and work simultaneously with the data. also its easy because I feel Python handles BIGDATA better than Perl. Also with the various packages and its interface with R to do statistics Python is what i prefer and feel is better. But I would prefer Bioperl over BIopython. »

Question : Mettre à jour ces déclarations à la date actuelle.

Perl, Java, C

Tester le programme *Perl* suivant de calcul de la plus grande sous-chaîne commune entre deux chaînes de caractères (Longest Common Substring)

```
$perl LCS.prl "coucou" "corageco"

$perl LCS.prl "AGGGTTTTGGGAA" "AGCCCCGTTAT"
```

Compiler le code *LCS.c*

```
$gcc -c LCS.c
$gcc -o LCS LCS.o
```

et exécuter le :

```
$./LCS "coucou" "corageco"
```

Compiler le code *LCS.java*

```
$javac LCS.java
```

et exécuter le

```
$java LCS
```

Comparer les temps d'exécution :
en *perl*,

```
# start timer
$start = time();
# perform a math operation 200000 times
...
# end timer
$end = time();
# report
print "Time taken was ", ($end - $start), " seconds";
```

En C, Java ?
https://fr.wikiversity.org/wiki/Fonctions_de_base_en_langage_C/time.h
En Java ?

```
long begin = System.currentTimeMillis();
...
long end = System.currentTimeMillis();
float time = ((float) (end-begin)) / 1000f;
```

Pour cela générer une courbe avec en abscisse la taille cumulée des séquences et en ordonnées les temps d'exécution.

La bibliothèque BioJava pour l'Alignement de Séquences

L'alignement de séquence de type Needleman-Wunsch (NW) ou Smith-Waterman (SW) utilise le même principe que la programmation dynamique avec des fonctions de coût différentes (*mismatch,* **match** et *gap (-)*) pour remplir les scores (valeurs des cases) et obtenir un alignement plus renseigné.
Récupérer l'archive du projet ***SequenceAlignement.zip*** dans le répertoire BioJava sur mon site et importer le dans Eclipse. Testez le en consultant le fichier *Usage.txt*
Il faudra faire le lien avec les bibliothèques *.jar* nécessaire disponibles sur mon site.

Et Python dans tout ça

La plupart des gros serveurs de logiciels bio-informatiques est écrit en C ou C++. BLAST était à l'origine écrit en C et maintenant il existe une version C++. Mais la plupart des petites applications écrites par les chercheurs (biologiste, bio-informaticien, biologiste computationnel) sont écrits en Perl. Sans doute car la plus grande base de ressources bio-informatiques, Bioperl, est écrite en Perl. Mais Python (et BioPython) gagne du terrain de façon exponentielle.

Question : Recoder cet algorithme en Python