par Carlos Andrés Pérez
<caperez/at/usc.edu.co>

L´auteur:
Carlos Andrés Pérez est spécialiste en simulation moléculaire, candidat au doctorat en Biotechnologie. Il est conseiller technique du Grupo de Investigación en Educación Virtual (GIEV) – Groupe de Recherche en Apprentissage Virtuel. Son adresse est : Universidad Santiago de Cali, Calle 5ª carrera 62 Campus Pampalinda, Cali – Colombia.

Traduit en Français par:
Jean–Etienne Poirrier
(homepage)

Simulation d'ADN assistée par ordinateur, en utilisant Linux et Perl

Résumé:

Cet article montre une manière de générer « n » séquences d'ADN avec « s » nucléotides, de manière aléatoire en utilisant des programmes Perl. Par contraste avec d'autres programmes où les séquences d'ADN sont traîtées comme des chaînes de texte, cet article propose de travailler avec ces séquences sous forme de tableaux. Même avec la génération aléatoire de séquences, lorsque nous effectuons l'analyse statistique de la fraction de nucléotides identiques placés dans la même position comparée, nous obtenons des valeurs similaires à 0.25.

Simulation par ordinateur

La recherche scientifique est focalisée sur l'étude et la compréhension des processus naturels. Les recherches sont faites en utilisant l'observation expérimentale et la modélisation théorique qui, depuis longtemps, ont été dédiées à l'exposition et au développement d'équations. Mais nombre de ces équations ne peuvent pas être résolues de manière analytique ou numérique à cause d'aspects techniques ou de l'impossibilité de représenter des systèmes naturels.

Le développement toujours croissant des sciences informatiques nous permet de simuler ces phénomènes naturels grâce à des logiciels qui peuvent remplacer les équations mathématiques pour décrire ces systèmes. Ces logiciels facilitent l'introduction de variables aléatoires qui facilitent la reproduction de processus biologiques et physiques dans l'ordinateur, la découverte d'aspects déterministes dans les simulations qui peuvent, peu après, être transformés en lois.

Pourquoi Perl ?

Perl (Practical Extraction and Report Language) est un langage structuré qui manipule des données sous forme de scalaire, liste, tableau, hashes et types de fichiers. Sa mise en application est déjà souple, parce qu'il est simple de générer des fonctions qui peuvent être évaluées avec n'importe quel types de données, outre ses capacités d'intégration aux bases de données, aux sites web dynamiques et systèmes d'exploitation comme Linux, en assistant l'utilisateur final dans ses tâches courantes.

Perl est très attractif comme outil analytique parce que sa syntaxe nous permet de travailler avec des structures très similaires à Mathematica, un langage symbolique puissant (http://www.xahlee.org/PerlMathematica_dir/perlMathematica.html) et que ses modules peuvent être intégrés avec les bibliothèques C, GTK, CGI, GGL, ajoutant ainsi des capacités d'intégration de programmes en un seul langage.

Il y a de nombreux programmes ayant de gros avantages en manipulation des données, structurée en vue d'une programmation mathématique. Mais ce sont des suites logicielles commerciales et cela limite leur dissémination et leur amélioration. En même temps, ils sont très restrictifs lorsqu'ils communiquent avec d'autres langages.

Perl ne limite pas la taille des données : cela dépend des capacités de mémoire et des ressources récursives. Le nombre de tables de hashage utilisées en tableaux associatifs n'est pas non plus restreint.

Dans cet article, chaque chaîne d'ADN a été représentée comme un vecteur. Nous pouvons utiliser le module PDL (Perl Data Language, http://pdl.sourceforge.net/WWW−old/index_es.html) qui est orienté vers le traitement numérique de données contenues dans des matrices de n dimensions. Mais nous avons des problèmes à définir chaque élément de vecteur comme un nucléotide car la fonction « pdl » ne prend en charge que des ordres numériques. Pour cette raison, nous n'utiliserons que les fonctions de base de Perl pour manipuler les tableaux. De toutes façons, cela ne limite pas les possibilités de représenter chaque nucléotide par un nombre.

Séquences aléatoires

Le programme nous permet de générer « n » chaînes d'ADN composées de « s » nucléotides. Chaque chaîne a la même longeur et ses composant sont des nucléotides choisis au hasard. Le programme convertit la liste de scalaires passés en paramètres par la fonction Dumper en une chaîne Perl qui décrit la structure des données. Ainsi, nous pourrons visualiser chaque vecteur.

```
#!/usr/bin/perl
# Use the module Data::Dumper
use Data::Dumper;
# $var1 contient les séquences de nombre, $var2 la longueur de la
# chaîne
print "Entrez le nombre de séquences aléatoire à générer\n";
$var1 = <STDIN>;
print "Entrez le nombre de nucléotides par séquence\n";
$var2 = <STDIN>;
# Dans aleatorio, nous définissons le tableau @ns qui contient les nucléotides,
# $lon est la variable locale qui stocke le nombre de nucléotides
# $col est la variable locale qui stocke le nombre de séquences
sub aleatorio {
local @ns = (a,g,c,t);
local $lon = $_[1];
```

```
local $col = $_[0];
# Nous définissons un tableau vide @a pour stocker les vecteurs.
@a = ();
# Les variables qui indiquent les vecteurs et les positions de ses composants.
local $i = 0;
local $u = 0;
while ($u <= $col - 1 && $i <= $lon - 1) {
# $result est la variable qui stocke l'élection aléatoire de
# chacun des nucléotides.
$result = @ns[int(rand(4))];
# Ajoute des nucléotides et stocke les chaînes qui les contiennent.

$a[$u][$i] = $result;
$i = $i + 1;
if ($i == $lon ) {
$u = $u + 1;
$i = 0;
}
}
return @a;
}
#Montre à l'écran chaque vecteur et ses composants.
print Dumper(&aleatorio($var1,$var2));
# Défini les positions et vecteurs initiaux utilisés pour comparer
# s'ils ont des nucléotides identiques aux mêmes positions.
$k = 0;
$count = 0;
$s1 = 0;
$s2 = 1;
while ($s1 <= $col - 2  &&  $s2 <= $col - 1 && $k <= $lon - 1 ) {
# S'ils sont identiques, $count est incrémenté de 1.
if ($a[$s1][$k] eq $a[$s2][$k]) {
$count = $count + 1;
}
# $k indique les nucléotides dans le vecteur, $s1 et $s2 sont les
# vecteurs comparés.
# Si la valeur de $k est la même, il y aura un drapeau à la
# position du nucléotide indiquant que la comparaison est finie.
$k = $k + 1;
if($k == $lon ) {
$k = 0;
$s2 = $s2 +1;
}
# Si $s2 est le même que $col, nous savons qu'un des vecteurs a
# été comparé aux autres
if ($s2 == $col ) {
$k = 0;
$s1 = $s1 + 1;
$s2 = $s1 + 1 ;
}
}
# Nous déterminons $pvalue qui montre le nombre de comparaisons.
for ($p = $col - 1, $r = $col -2; $r >= 0 ; $r--){
$p+= $r;
}
# Les résultats sont montré.
print "Le nombre de nucléotides identiques est : $count\n";
print "Le nombre de comparaisons est : $p\n";
$y = $count/$p;
# Dans $cor, nous stockons la valeur de la fraction de nucléotides
# identiques dans la même position durant la comparaison
$cor = $y/$lon;
print "La fraction de nucléotides apparaissant dans la même position est : $cor";
```

En estimant la valeur $cor pour 10 séquences de 30, 50, 70 et 90 nucléotides, nous obtenons les résultats suivants : 0.2340, 0.26, 0.26031, 0.2661.

Pour 40 séquences of 100, 200, 300 et 500 nucléotides, nous obtenons les valeurs suivantes de $cor : 0.2507, 0.2482, 0.2480, 0.2489.

En conséquence, nous pouvons donc conclure que pour un nombre « n » de séquences d'ADN avec « s » nucléotides générés de manière aléatoire, la fraction de nucléotides identiques qui sont dans la même position est approximativement 0.25.

Bibliographie

- http://bioperl.org/
- http://pdl.perl.org/index_es.html
- http://www.unix.org.ua/orelly/perl/prog3/
- http://www.xahlee.org/PerlMathematica_dir/Matica.html
- Exemples de manipulation de fichiers en Perl créé par le Dr. Antonio J. Pérez Pulido pour la maîtrise en Bioinformatique de l'Andalucía International University.
- article 374 d'avril 2005 sur linuxfocus.org

Téléchargement de fichiers

code source Perl : ADNaleatorio_pl.txt