# BioinformaticsStronghold

Alejandro Cebrián del Valle

18/09/2020

## Contando Nucleótidos...

En este caso, se compara una simple cuenta de nucleótidos. La forma más simple que encuentro, es simplemente crear un diccionario de cada una de las letras. Puede hacerse mediante el método de la clase **String**. En este caso, para que con unos simples cambios permitan hacerlo con k-meros que no se solapen, **Biopython** permite crear un objeto *secuencia*, que permite usarlo. Por lo que vemos, es más rápido Biopython (quizás, tenga que ver que no tiene que recorrer toda la cadena ;) (como el método count())).

```
from Bio.Seq import Seq
from time import time
### Counting Nucleotides
def countingNucleotides(sequence):
    nucleotides = {}
    for nucl in sequence:
        if nucl not in nucleotides.keys():
            nucleotides[nucl] = 1
        else:
            nucleotides[nucl] += 1
    return nucleotides

### The program
sequence = "CTACATGCCGACATCCATGCCGTAGTGACTATTTTATGATAGGTGTTCGGACATCCACCA" + \
           "CAATCTGACCAGCGAGTTCGGGCTTCCTTCAACTAATCTTGAATCCATCATCAACCTCGT" + \
           "GTTAATAGAACGAATTATTTCGTACTTACTTCACCCTAGCCCTTATGGTTTACCCGGCAC" + \
           "CTGACTAAAGCTAACGCCCTACTCGCCTAGCCTGCCCATTAACCCCATTCGATGCAGATG" + \
           "CAACTATACTGTGCCTAGCGAAACTCGCCCGCTGGCATGAAAAGAACATAGTCCTTACCT" + \
           "TAGTGAGCGCACGACACGATCACACATCGAGGGTGGTCGGTTTGTCTAGGAGCGCGTGAC" + \
           "AATAAAATTTTCTCATAGTTCTATGTTTTGCGGCACTTAGCTTTCGCCCCTCGGTGCGCG" + \
           "AGAATTCTAGACCATGCGTAATCAAAAACTTCTGGGAAGAAGCACCTAAATAATCCAAAA" + \
           "ACTAACACCTGGTACAAACTTAGCAATACACAGGCTAGACGATTCGTAGAATGCCTATCG" + \
           "GTCACGGTAGAGCAATCATGTTATTTTAGCATCTCTATAAGTCTTTTTCGACGGCTACAT" + \
           "CGTTATATATGGGTATTGTCCCATCATTACGAAAATTGCGGCGGCCTGATATATTCTATT" + \
           "GGGCATATTGAGACAAAGCCATCAGAACTACGGCTTCGCATGTGTGGTCTACTATGTTGT" + \
           "AGGACCTGGGATAAACATAATCCTTGCTTCGTTTTTAAGTAGCAGCCGACTCAAACCCTG" + \
           "CCCTATCAGCAACAGTCCTCTGTTCCAGTTAAAGAAAGGTACAACAAGCTGATCAGTTAG" + \
           "GACGCCGGAAAACAAACTAAACGACCCGCGGTAATACCGGCGAAATTGGGGGAGGCGTCA" + \
           "ATGTTCTAAAGGGACTGAGATATAGAAACGACTAACCAGTAGGTTCACACGACTCACGTG" + \
           "TTATGCTTAGTTCTAGCGACT"
time1 = time()
print(" ".join([str(i) for i in countingNucleotides(sequence).values()]))
```

```
## 243 258 277 203
```

```
time2 = time()
###### The biopythonic way
bioSequence = Seq(sequence)
nucleotideList = [bioSequence.count("A"),
                  bioSequence.count("C"),
                  bioSequence.count("G"),
                  bioSequence.count("T")]
print(" ".join([str(x) for x in nucleotideList]))
```

```
## 277 243 203 258
```

```
time3 = time()
print("The diference between the two times:\n" + \
      str(time2 - time1) + " (The normal way)\n" + \
      str(time3 - time2) + " (The biopythonic way)")
```

```
## The diference between the two times:
## 0.010317564010620117 (The normal way)
## 0.041934967041015625 (The biopythonic way)
```

En el caso de R, aunque **Bioconductor** se establece como un buen repositorio de paquetes de software para bioinformática, no deja de ser un repositorio y no posee las particularidades que tiene **Biopython** como paquete en sí. No obstente, la forma más optima es convertir la secuencia en un vector de caracteres (lo que hace la función *strsplit*) para posteriormente tabularla en un formato Caracter:Repeticiones del carácter. Aunque R también permite recorrer la cadena mediante bucle *for*, este se prefiere eliminar por la mayoría de los programadores en R.

```
seqRossalind = paste("CTACATGCCGACATCCATGCCGTAGTGACTATTTTATGATAGGTGTTCGGA" ,
                     "CATCCACCACAATCTGACCAGCGAGTTCGGGCTTCCTTCAACTAATCTTGA" ,
                     "ATCCATCATCAACCTCGTGTTAATAGAACGAATTATTTCGTACTTACTTCA" ,
                     "CCCTAGCCCTTATGGTTTACCCGGCACCTGACTAAAGCTAACGCCCTACTC" ,
                     "GCCTAGCCTGCCCATTAACCCCATTCGATGCAGATGCAACTATACTGTGCC" ,
                     "TAGCGAAACTCGCCCGCTGGCATGAAAAGAACATAGTCCTTACCTTAGTGA" ,
                     "GCGCACGACACGATCACACATCGAGGGTGGTCGGTTTGTCTAGGAGCGCGT" ,
                     "GACAATAAAATTTTCTCATAGTTCTATGTTTTGCGGCACTTAGCTTTCGCC" ,
                     "CCTCGGTGCGCGAGAATTCTAGACCATGCGTAATCAAAAACTTCTGGGAAG" ,
                     "AAGCACCTAAATAATCCAAAAACTAACACCTGGTACAAACTTAGCAATACA" ,
                     "CAGGCTAGACGATTCGTAGAATGCCTATCGGTCACGGTAGAGCAATCATGT" ,
                     "TATTTTAGCATCTCTATAAGTCTTTTTCGACGGCTACATCGTTATATATGG" ,
                     "GTATTGTCCCATCATTACGAAAATTGCGGCGGCCTGATATATTCTATTGGG" ,
                     "CATATTGAGACAAAGCCATCAGAACTACGGCTTCGCATGTGTGGTCTACTA" ,
                     "TGTTGTAGGACCTGGGATAAACATAATCCTTGCTTCGTTTTTAAGTAGCAG" ,
                     "CCGACTCAAACCCTGCCCTATCAGCAACAGTCCTCTGTTCCAGTTAAAGAA" ,
                     "AGGTACAACAAGCTGATCAGTTAGGACGCCGGAAAACAAACTAAACGACCC" ,
                     "GCGGTAATACCGGCGAAATTGGGGGAGGCGTCAATGTTCTAAAGGGACTGA" ,
                     "GATATAGAAACGACTAACCAGTAGGTTCACACGACTCACGTGTTATGCTTA" ,
                     "GTTCTAGCGACT", sep  = "")
table(strsplit(seqRossalind, "")[[1]])
```

```
##
##   A   C   G   T
## 277 243 203 258
```