Bioinformática avanzada: problemas y algoritmos

SESIÓN 3

Rodrigo Santamaría

2014

¿Genética o estadística?

MUTACIONES

Mutaciones

- Las mutaciones del código genético son
  - Una realidad **biológica**
  - Una complicación **computacional**
  - Un problema **estadístico**

Permutaciones con repetición

- ¿Cuántas posibles cadenas de DNA de longitud n existen?
  - Tenemos r elementos (los 4 nucleótidos)
  - Hacemos n elecciones
  - Permutaciones con repetición: n^r
  - Para una cadena de 30 nucleótidos
    - 4^{30}=1152921504606846976 posibles cadenas!

Ejercicio

- Implementar la función `mutations`:
  - **entrada**
    - `word`: palabra de la que buscamos todas la mutaciones
    - `letters`: posibles letras en word o en las mutaciones
    - `num_mismatches`: número de mutaciones puntuales
  - **salida**
    - vector con todas las posibles mutaciones de `word`
- Implementar `mutationsEqualOrLess` que calcule todas las mutaciones con *hasta* `num_mismatches`

Solución

```
def mutations(word, letters, num_mismatches):
    import itertools
    for locs in itertools.combinations(range(len(word)), num_mismatches):
        this_word = [[char] for char in word]
        for loc in locs:
            orig_char = word[loc]
            this_word[loc] = [l for l in letters if l != orig_char]
        for poss in itertools.product(*this_word):
            yield ''.join(poss)
```

```
def mutationsEqualOrLess(word, num_mismatches, letters="ACGT"):
    matches=set()
    for dd in range(num_mismatches,-1,-1):
        matches.update(list(mutations(word, letters, dd)))
    return matches
```

Pues va a ser que no nos libramos de la estadística

COMBINACIONES Y PERMUTACIONES

Combinaciones y permutaciones

- **Combinación**: mezcla sin importar el orden
- **Permutación**: mezcla *ordenada*
  - *__p__ermutación – __p__osición*
- En ambos casos, podemos permitir repetición de elementos o no

- Combinación: 473, 374, 347 son lo mismo
- Permutación: 473 es distinto de 374 y 347

Permutaciones con repetición

- ¿Cuántas posibles cadenas de DNA de longitud n existen?
  - Tenemos r elementos (los 4 nucleótidos)
  - Hacemos n elecciones: r·r·r…·r (n veces)
  - Permutaciones con repetición: n^r
  - Para una cadena de 30 nucleótidos
    - 4^{30}=1152921504606846976 posibles cadenas!

Permutaciones sin repetición

- Tenemos *n* elementos entre los que elegir *r*
  - El nº de permutaciones con repetición posibles es
    - n x (n-1) x (n-2) … x (n-r)
    - En otras palabras, hay una posibilidad menos con cada nueva elección

- Función factorial
  - n! = n x (n-1) x (n-2) x … x 3 x 2 x 1
  - 4! = 4 x 3 x 2 x 1 = 24

Permutaciones sin repetición

- Sean los nucleótidos A, T, C, G (n=4)
- Las posibles permutaciones sin repetición de cadenas de longitud r=4 serán:

```
A → T → C → G
    C   G
    G
T → A → C → G
    C   G
    G
…
4 · 3 · 2 · 1 = 24
```

Permutaciones sin repetición

- Podemos tomar cadenas de longitud r como mucho igual al número de elementos n
- Si es menor, la probabilidad no es el factorial
  - n x (n-1) x (n-2) … x (n-r)

$$\frac{n!}{(n-r)!}$$

Combinaciones sin repetición

- En una combinación no importa el orden
- La mejor forma de verlo es tomar todas las posibles permutaciones sin repetición (r=n)
- Por ejemplo, para r=n=3

Importa el orden (3!)	No importa el orden (1)
1 2 3	
1 3 2	
2 1 3	123
2 3 1	
3 1 2	
3 2 1	

Combinaciones sin repetición

- Por lo tanto, lo único que tenemos que hacer es reducir las permutaciones sin repetición por r!

$$\frac{n!}{r!(n-r)!} = \binom{n}{r}$$

- Esta es una función importante (coeficiente binomial) y se representa como se ve arriba

Combinaciones con repetición

- Esta fórmula es un poco más difícil de deducir intuitivamente:

$$\left(\begin{array}{c} n+r-1 \\ r \end{array} \right) = \frac{(n+r-1)!}{r!(n-1)!}$$

Resumen

Posición?	Repetición?	Posibilidades (n elems., r elecs.)	Comentarios
Permutación (sí)	Sí	n^r	n posibilidades por elección
	No	$\frac{n!}{(n-r)!}$	n posibilidades en la 1ª elección n-1 en la 2ª elección, etc.
Combinación (no)	Sí	$\frac{(n+r-1)!}{r!(n-1)!}$	
	No	$\frac{n!}{r!(n-r)!}$	Como la permutación sin repetición pero eliminando las posibles combinatorias en cada ronda (*r!*)

Más información: http://www.mathsisfun.com/combinatorics/combinations-permutations.html

Ejercicio

- Sea una cadena *S* de 11 nucleótidos
  - Cuantas cadenas posibles mutadas en 2 nucleótidos existen para *S*?
- **1º paso:**
  - Cuántas posibles combinaciones de lugares hay para mutaciones de *S* en 2 nucleótidos?
    1. Cuánto valen n y r?
       - r es el nº de cambios que queremos → 2
       - n es el nº de puntos posibles de cambio → 11
    2. Importa el orden de las mutaciones?
       - no (combinación)
    3. Hay posiciones mutadas 2 veces (repeticiones)?
       - no (sin repetición)

Ejercicio

$$\frac{11!}{2!(11-2)!} = 55$$

- Tenemos 55 posibles pares de posiciones donde puede haber mutación

Ejercicio

- **2º paso:** para cada uno de esos pares, ¿cuántas posibles mutaciones puede haber?
  - r, puntos de cambio → 2
  - n, opciones de cambio → 3
  - Importa el orden? → Sí (permutación)
  - Hay repetición? → Sí
  - $3^2=9$
- $55 \cdot 9=495$ posibles mutaciones en 2 nucleótidos en una cadena de 11
- *Y todas las posibles mutaciones en 3 nucleótidos?*

Ahora sí que os vamos a pillar

MOTIVOS ESQUIVOS

Motivos esquivos

- El problema que teníamos antes es que sólo teníamos en cuenta motivos que aparecieran en nuestra secuencia
  - Pero puede haber motivos que ***no*** aparezcan en la secuencia pero por mutaciones sean los más comunes!

Ejercicio

- Implementar la función `allMutations`:
  - **entrada**
    - `seq`: secuencia de la que buscamos todos los k-mers
    - `k`: tamaño de los k-mers
    - `d`: número de mutaciones puntuales
  - **salida**
    - vector con todas los `k`-mer posibles en `seq`, incluidos aquellos que no se encuentran explícitamente en `seq`, considerando hasta `d` mutaciones puntuales

Ejercicio

- Pista:
  - Utilizar la función `mutationsEqualOrLess` y una variable conjunto de tipo `set`
- Prueba:
  - seq: `CGCCCGAATCCAGAACGCATTCCCATATTTCGGGAC CACTGGCCTCCACGGTACGGACGTCAATCAAAT`
  - k: `9`
  - d: `3`
  - salida: 20847 mutaciones posibles distintas

Ejercicio

- Modificar ligeramente la función anterior para hacer la función `mostFrequentKmersV2`:
  - **entrada**:
    - `seq` (cadena donde buscamos)
    - `d` (nº de mutaciones permitidas)
    - `k` (tamaño del k-mero)
  - **salida**: lista con los `k`-mers más frecuentes en `seq`, contando hasta `d` mutaciones, teniendo en cuenta `k`-mers que no están en la secuencia

Ejercicio

- Prueba:
  - seq: `AACAAGCTGATAAACATTTAAAGAG`
  - k: `5`
  - d: `1`
  - salida: `AAAAA`

Ejercicio

- Modificar ligeramente la función anterior para hacer la función `countKmersMMV2`:
  - **entrada**:
    - `seq` (cadena donde buscamos)
    - `d` (nº de mutaciones permitidas)
    - `k` (tamaño del k-mero)
    - `min` (nº mínimo de veces que se encuentra el k-mer)
  - **salida**: diccionario con los `k`-mers en `seq` (clave) y la frecuencia con la que ocurren (valor), contando hasta `d` mutaciones, teniendo en cuenta `k`-mers que no están en la secuencia

Ejercicio

- Prueba:
  - seq: `AACAAGCTGATAAACATTTAAAGAG`
  - k: `5`
  - d: `1`
  - min: `3`
  - salida: `{'AACAG': 3, 'TAAAA': 3, 'ATTAA': 3, 'AAGAT': 3, 'AAAAA': 4, 'TTAAA': 3, 'AAAAG': 3}`

Buscando DNA boxes

- Recordemos que la DNA box conocida experimentalmente en *E coli* es `TTACCACA`, que aparece 4 veces en la región oricEC*
- Si ejecutamos `mostFrequentKmersV2`
  - `k=9, d=1`
  - Obtenemos `['ATAATGATC', 'ATAATGAAC']`
- Si ejecutamos `countKmersMMV2`
  - `TTACCACA` tiene 2 ocurrencias
  - Los k-mers más frecuentes tienen 3
  - ¿Qué falla?

* http://en.wikipedia.org/wiki/Prokaryotic_DNA_replication

Ejercicio

- Modificar ligeramente la función anterior para hacer la función `countKmersMMRC`:
  - **entrada**:
    - `seq` (cadena donde buscamos)
    - `d` (nº de mutaciones permitidas)
    - `k` (tamaño del k-mero)
    - `min` (nº mínimo de veces que se encuentra el k-mer)
  - **salida**: diccionario con los `k`-mers en `seq` (clave) y la frecuencia con la que ocurren (valor), contando hasta `d` mutaciones, teniendo en cuenta `k`-mers que no están en la secuencia y las ocurrencias del k-mer reverso compementario

Ejercicio

- Prueba:
  - seq: `AACAAGCTGATAAACATTTAAAGAG`
  - k: `5`
  - d: `1`
  - min: `4`
  - salida:
```
{'AAAAA': 4, 'CTTTT': 4,
'AACAG': 4, 'TTTAA': 5, 'TAAAA': 5,
'TTAAT': 4, 'TTGAA': 4, 'GTTAA': 4,
'ATTAA': 4, 'TTCAA': 4, 'TTAAC': 4,
'TTATA': 4, 'TTAAA': 5, 'TTTTA': 5,
'TATAA': 4, 'CTGTT': 4, 'AAAAG': 4}
```

Buscando DNA boxes

- Contando las ocurrencias inversas complementarias, sí que encontramos las cuatro ocurrencias de TTATACACA:

AATGATGATGACGTCAAAAGGATCCGGATAAAACATGGTGATTGCCTCGCATAACG
CGGTATGAAAATGGATTGAAGCCCGGGCCGTGGATTCTACTCAACTTTGTCGGCTT
GAGAAAGACCTGGGATCCTGGGTATTAAAAAGAAGATCTATTTATTTAGAGATCTG
TTCTATTGTGATCTCTTATTAGGATCGCACTGCCC**TGTGGATAA**CAAGGATCCGGC
TTTTAAGATCAACAACCTGGAAAGGATCATTAACTGTGAATGATCGGTGATCCTGG
ACCGTATAAGCTGGGATCAGAATGAGGGG**TTATACACA**ACTCAAAAACTGAACAAC
AGTTGTTC**TTTGGATAA**CTACCGGTTGATCCAAGCTTCCTGACAGAG**TTATCCACA**
GTAGATCGCACGATCTGTATACTTATTTGAGTAAATTAACCCACGATCCCAGCCAT
TCTTCTGCCGGATCTTCCGGAATGTCGTGATCAAGAATGTTGATCTTCAGTG

Buscando DNA boxes

- Hemos tenido mucha suerte de que el motivo estuviera justo en los 500 nucleótidos que hemos elegido
- Además, hay otros motivos con 4 ocurrencias en esta zona

Conclusiones

- Aunque las predicciones computacionales son muy poderosas, hace falta colaboración de bioinformáticos y biólogos para verificarlas
- Aún así, incluso dar una pequeña lista de 9-mers con los que trabajar como candidatos a DNA boxes (o cualquier otro motivo) es una grandísima ayuda para un biólogo

Detalle de la portada de la versión japonesa del libro
“An introduction to bioinformatics algorithms”, de N. C. Jones y P. Pevzner (http://bix.ucsd.edu/bioalgorithms/)