Origen de replicación

Bioinformática 2025-2

Universidad de Sonora

8 de octubre de 2025

¿Cómo encontrar un gato negro en un cuarto oscuro?

53‡‡†305))6·;4826)4‡.)4‡);806·;48†8^60))85;161;:‡·8

†83(88)5·†;46(;88·96·?;8)·‡(;485);5·†2:·‡(;4956·2(5

·—4)8^8·;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4

)485†528806·81(‡9;48;(88;4(‡?34;48)4‡;1‡(;:188;‡?;

¿Cómo encontrar un gato negro en un cuarto oscuro?

53‡‡†305))6·;4826)4‡.)4‡);806·;48†8^60))85;161;:‡·8

†83(88)5·†;46(;88·96·?;8)·‡(;485);5·†2:·‡(;4956·2(5

·—4)8^8·;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4

)485†528806·81(‡9;48;(88;4(‡?34;48)4‡;1‡(;:188;‡?;

¿Cómo encontrar un gato negro en un cuarto oscuro?

53‡‡†305))6·THE26)H‡.)H‡)TE06·THE†E^60))E5T161T:‡·E

†E3(EE)5·†TH6(TEE·96·?TE)·‡(THE5)T5·†2:·‡(TH956·2(5

·—H)E^E·TH0692E5)T)6†E)H‡‡T1(‡9THE0E1TE:E‡1THE†E5TH

)HE5†52EE06·E1(‡9THET(EETH(‡?3HTHE)H‡T1‡(T:1EET‡?T

Contar k-meros

```
PatternCount(Text, Pattern)
    count ← 0
    for i ← 0 to |Text| − |Pattern|
        if Text(i, |Pattern|) = Pattern
            count ← count + 1
    return count
```

K-meros más frecuentes

```
FrequentWords(Text, k)
    FrequentPatterns ← an empty set
    Count ← an array of length |Text|-k+1
    for i ← 0 to |Text| - k
        Pattern ← Text(i, k)
        Count(i) ← PatternCount(Text, Pattern)
    maxCount ← maximum value in array Count
    for i ← 0 to |Text| - k
        if Count(i) = maxCount
            add Text(i, k) to FrequentPatterns
    remove duplicates from FrequentPatterns
    return FrequentPatterns
```

Algo más eficiente...

ACG	CGT	GTT	TTT	TTC	TCA	CAC	TTA	TAC	CGG
3	2	2	3	1	1	1	1	1	1

FrequencyTable

```
FrequencyTable(Text, k)
    freqMap ← empty map
    n ← |Text|
    for i ← 0 to n − k
        Pattern ← Text(i, k)
        if freqMap[Pattern] doesn't exist
            freqMap[Pattern]← 1
        else
            freqMap[pattern] ←freqMap[pattern]+1
    return freqMap
```

BetterFrequentWords

```
BetterFrequentWords(Text, k)
    FrequentPatterns ← an array of strings of length 0
    freqMap ← FrequencyTable(Text, k)
    max ← MaxMap(freqMap)
    for all strings Pattern in freqMap
        if freqMap[pattern] = max
            append Pattern to frequentPatterns
    return frequentPatterns
```

Resultados en el Ori de Vibrio Cholerae

atcaatgatcaacgtaagcttctaagcATGATCAAGgtgctcacacagtttatccacaac

ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca

cggaaagATGATCAAGagaggatgatttcttggccatatcgcaatgaatacttgtgactt

gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt

acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga

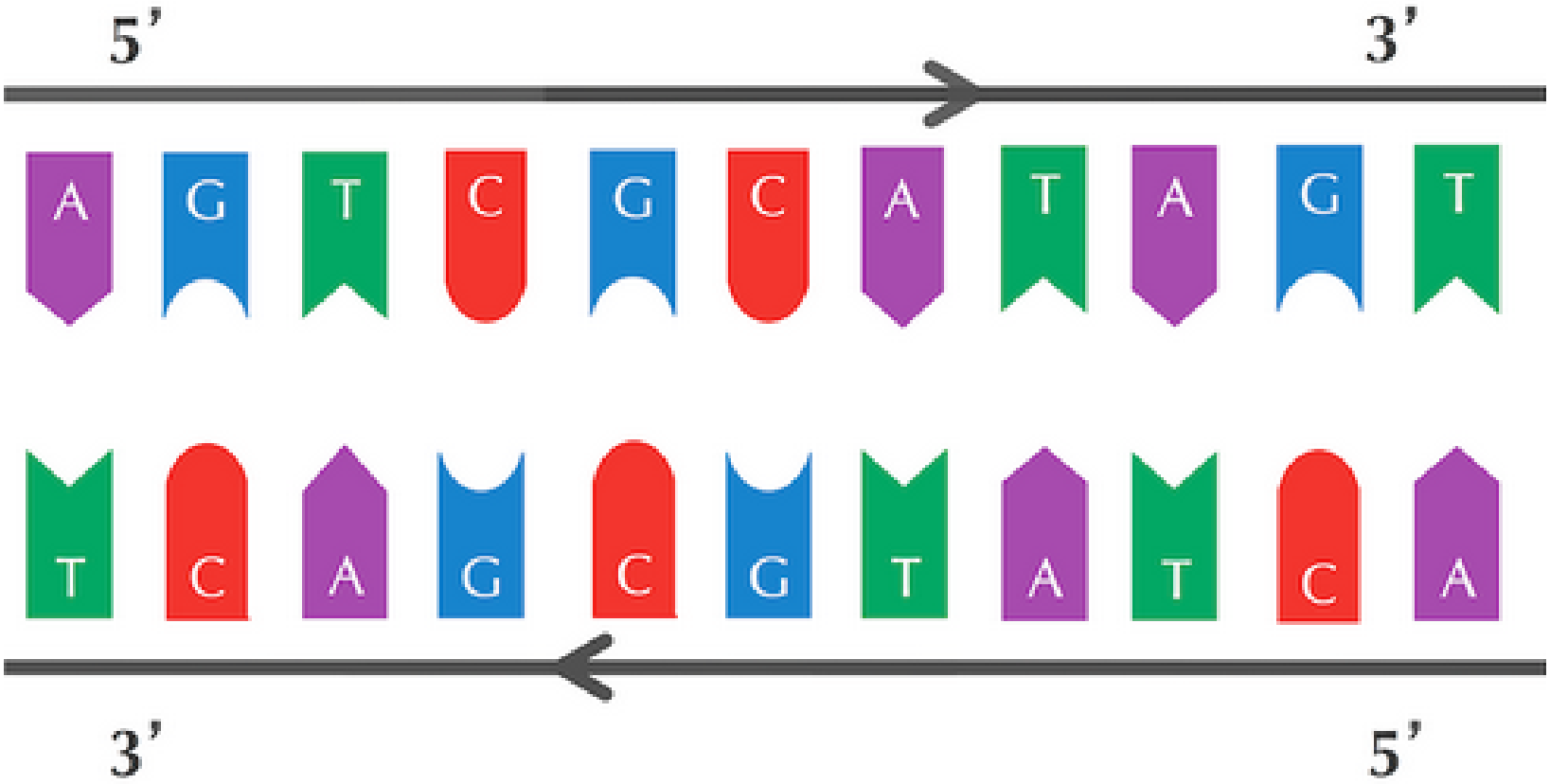
tagacggttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat

tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag

atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt

tccttaaccctctatttttacggaagaATGATCAAGctgctgctcttgatcatcgtttc

¿Podemos utilizar otro patrón?



Nuevamente, el Ori de Vibrio Cholerae

atcaatgatcaacgtaagcttctaagcATGATCAAGgtgctcacacagtttatccacaac

ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca

cggaaagATGATCAAGagaggatgatttcttggccatatcgcaatgaatacttgtgactt

gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt

acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga

tagacggttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat

tgataatgaatttacatgcttccgcgacgatttacctCTTGATCATcgatccgattgaag

atcttcaattgttaattctcttgcctcgactcatagccatgatgagctCTTGATCATgtt

tccttaaccctctatttttacggaagaATGATCAAGctgctgctCTTGATCATcgtttc

Nuevo problema: localización de los 9-meros.

No solamente nos interesa ahora saber el número de veces que aparece un k-mero, si no donde. Tenemos ahora un nuevo problema computacional:

"Dada una cadena *Genome* y un patrón *Pattern*, encontrar todas las posiciones en las cuales se presenta *Pattern*.

Otro caso: Thermotoga Petrophila

aactctatacctccttttttgtcgaatttgtgtgatttatagagaaaatcttattaactga

aactaaaatggtaggtttGGTGGTAGGttttgtgtacattttgtagtatctgatttttaa

ttacataccgtatattgtattaaattgacgaacaattgcatggaattgaatatatgcaaa

acaaaCCTACCACCaaactctgtattgaccattttaggacaacttcagGGTGGTAGGttt

ctgaagctctcatcaatagactattttagtctttacaaacaatattaccgttcagattca

agattctacaacgctgttttaatgggcgttgcagaaaacttaccacctaaaatccagtat

ccaagccgatttcagagaaacctaccacttacctaccacttaCCTACCACCcgggtggta

agttgcagacattattaaaaacctcatcagaagcttgttcaaaaatttcaatactcgaaa

CCTACCACCtgcgtcccctattatttactactactaataatagcagtataattgatctga

Un nuevo enfoque

Definición. Sean L, k dos números naturales. Decimos que un k-mero *Pattern* forma un (L, t)-grupo dentro de una cadena *Genoma* si existe una subcadena de longitud L dentro de *Genoma* en el cual aparece *Pattern* por lo menos t veces.

Ejemplo

El 4-mero TGCA forma un (25,3)-grupo en la siguiente cadena:

gatcagcataagggtccC**TGCA**A**TGCA**TGACAAGCC**TGCA**GTtgttttac

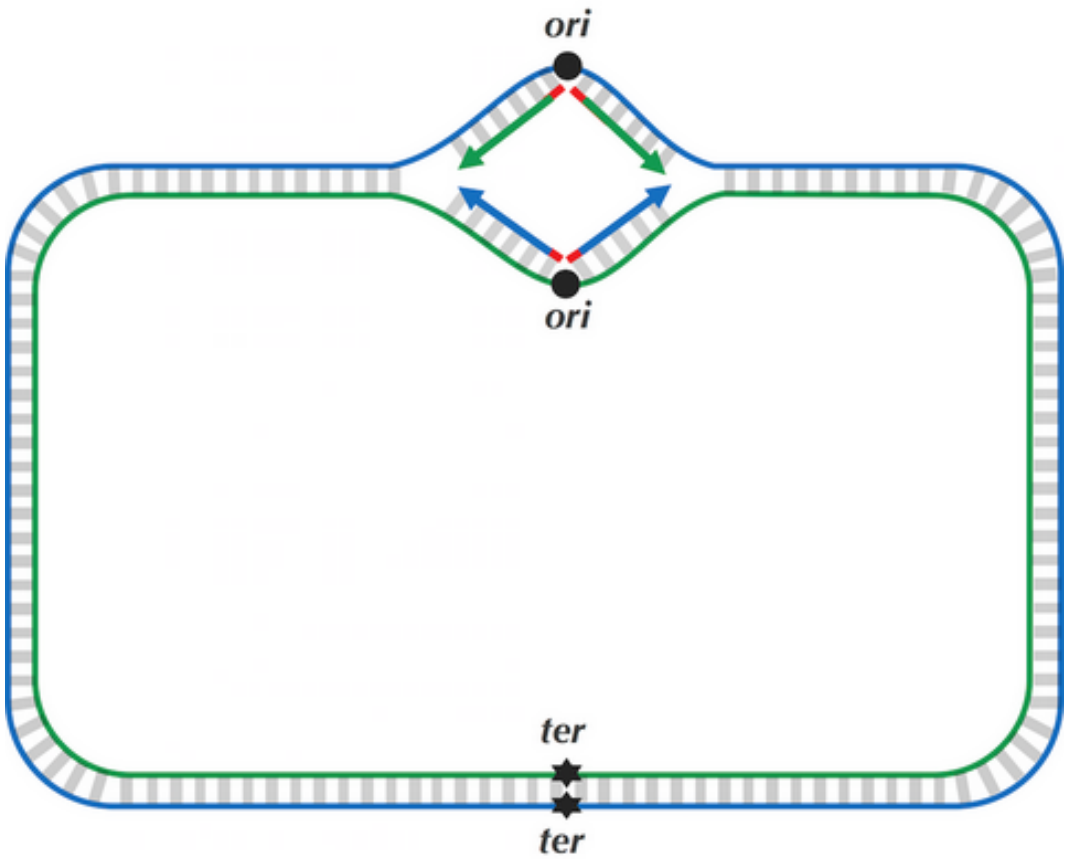
Nuevo problema computacional

Encontrar grupos. Dada una cadena *Genoma* y tres números naturales k, L y t, encontrar todos los distintos k-meros que forme (L, t)-grupos dentro de *Genome*

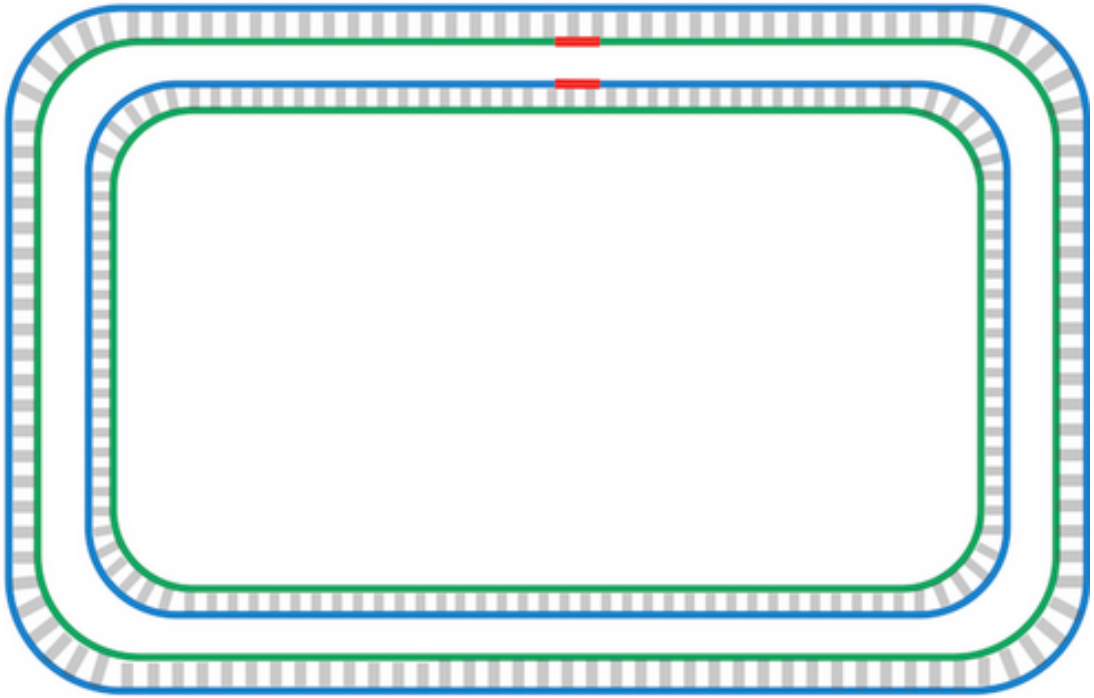
Algoritmo FindClumps

```
FindClumps(Text, k, L, t)
    Patterns ← an array of strings of length 0
    n ← |Text|
    for every integer i between 0 and n − L
        Window ← Text(i, L)
        freqMap ← FrequencyTable(Window, k)
        for every key s in freqMap
            if freqMap[s] ≥ t
                append s to Patterns
    remove duplicates from Patterns
    return Patterns
```

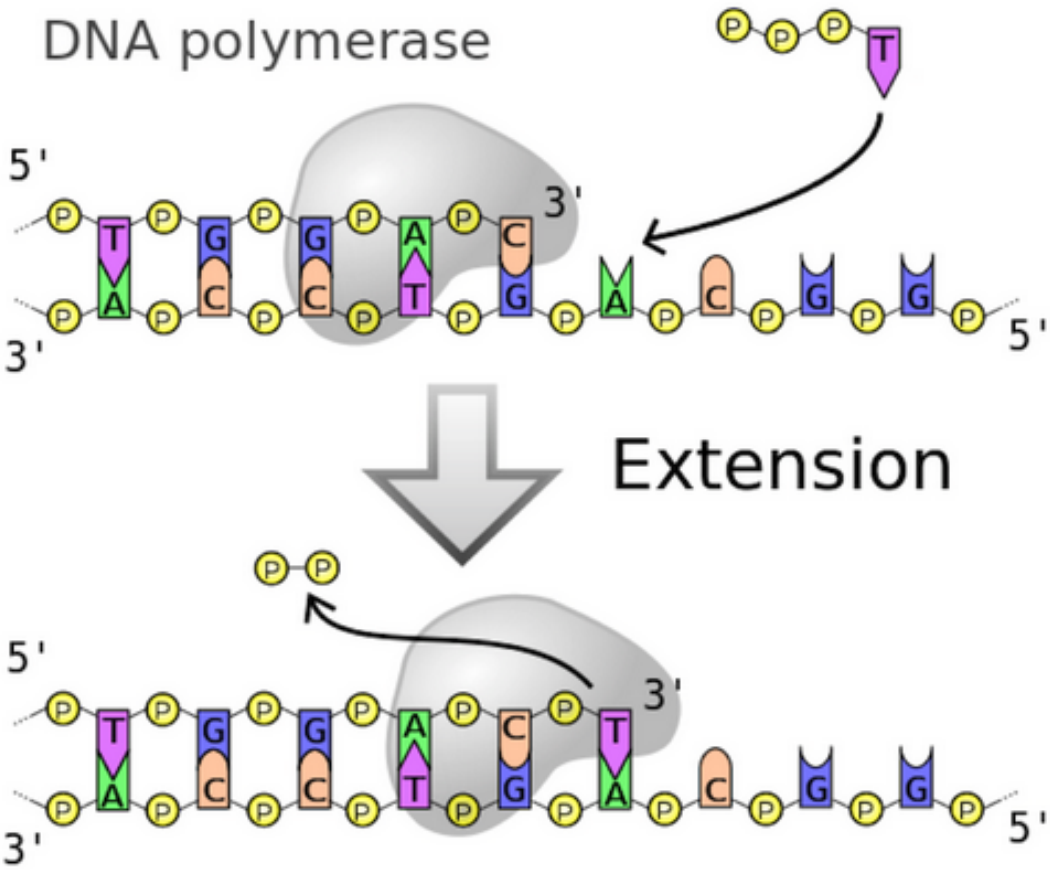
Recordando el proceso de replcación



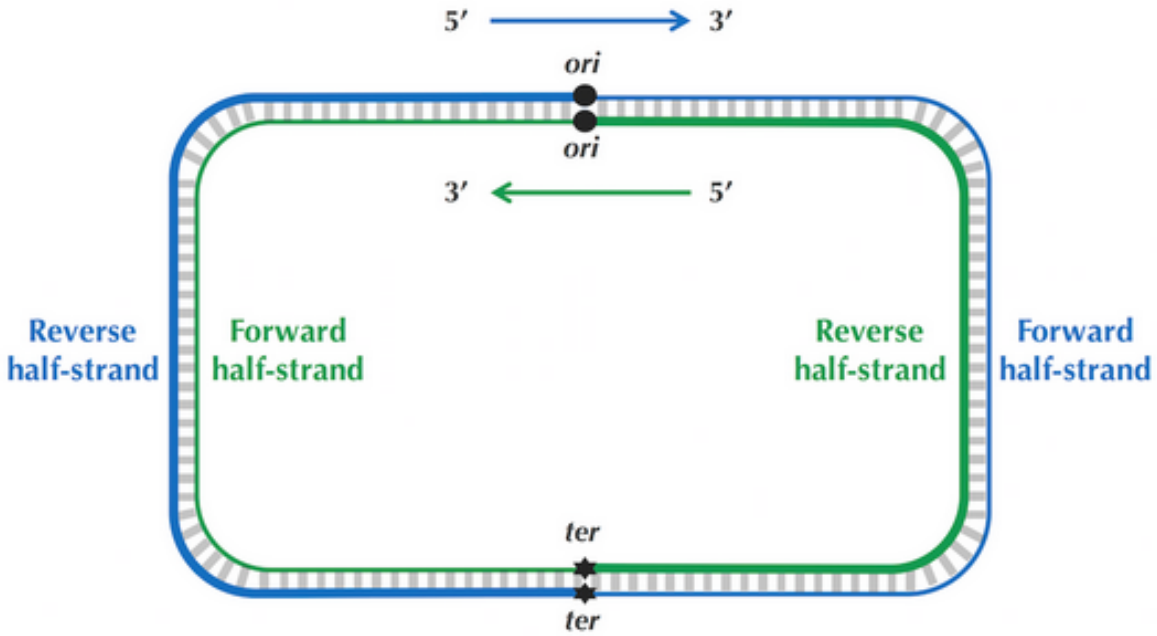
Recordando el proceso de replcación



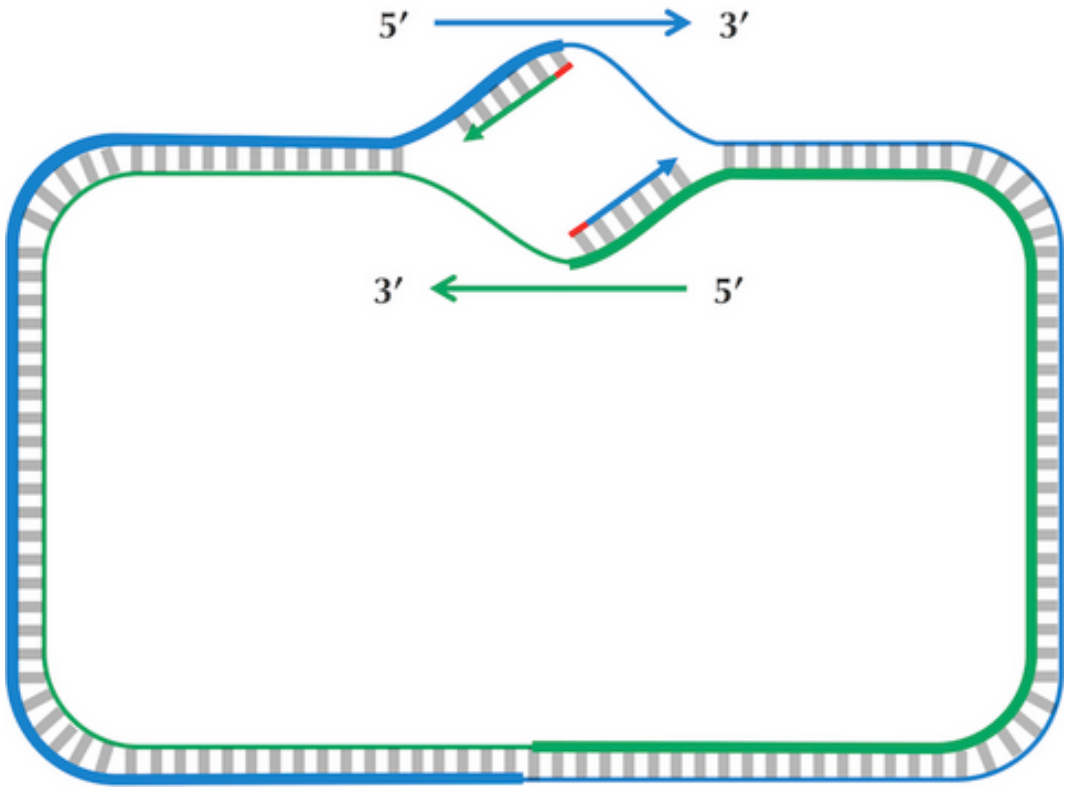
ADN Polimerasa



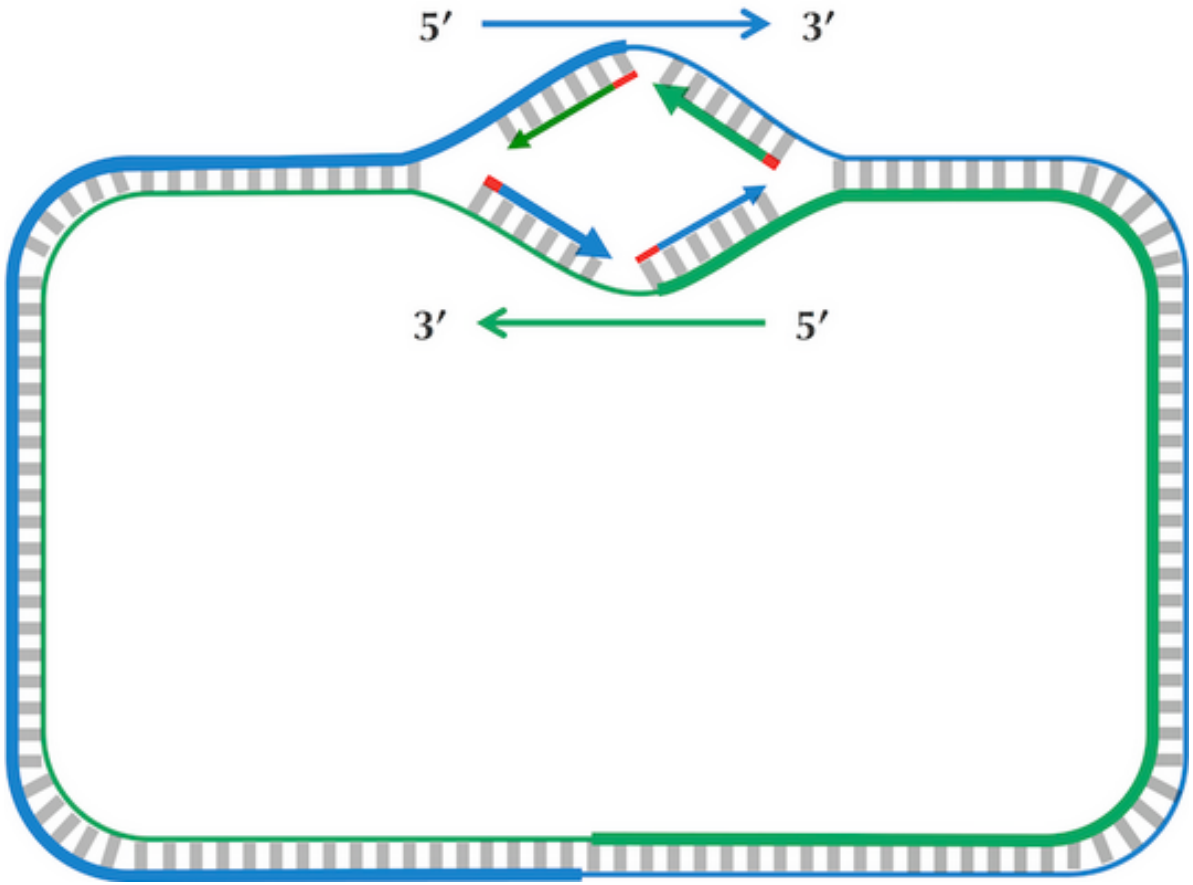
Proceso de replicación



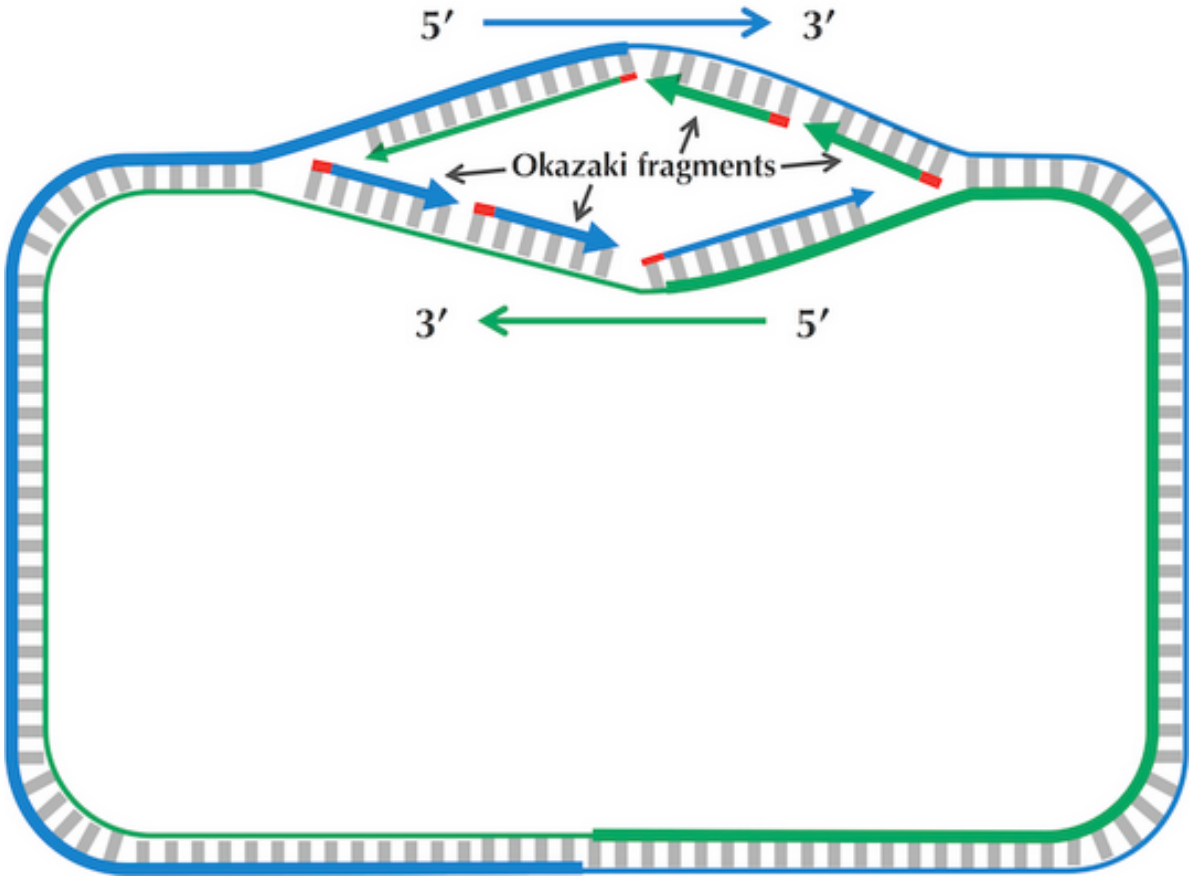
Proceso de replicación



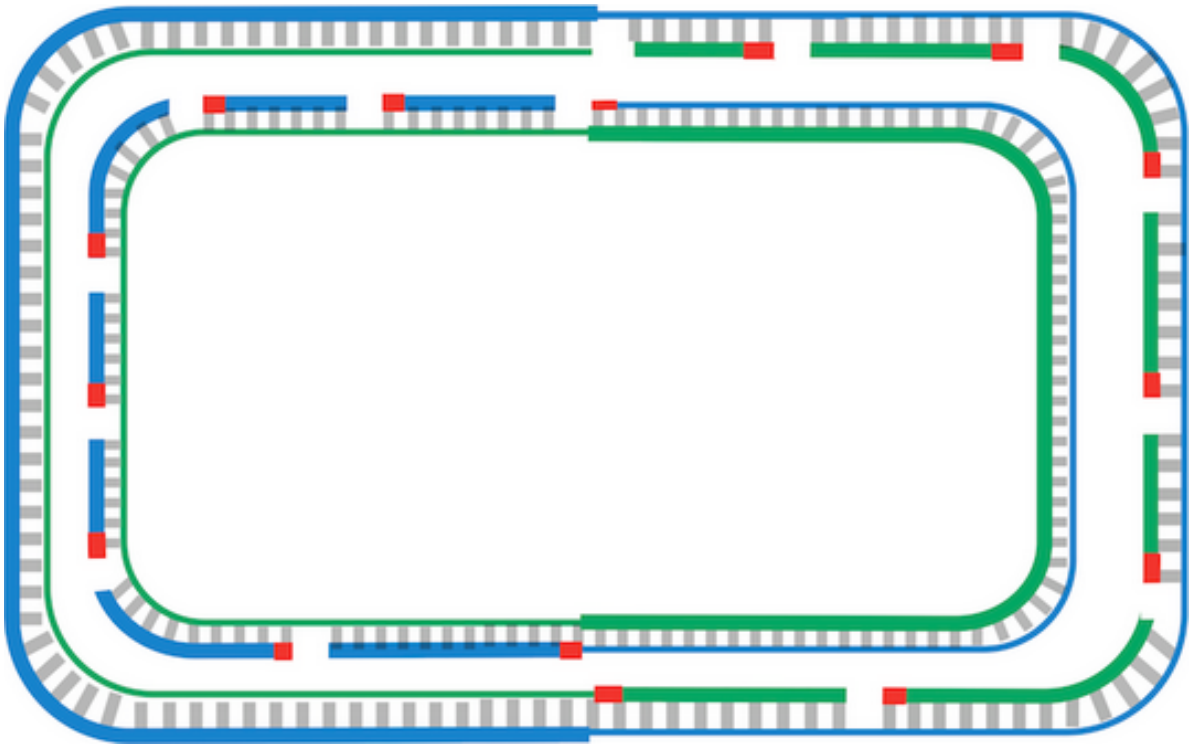
Proceso de replicación



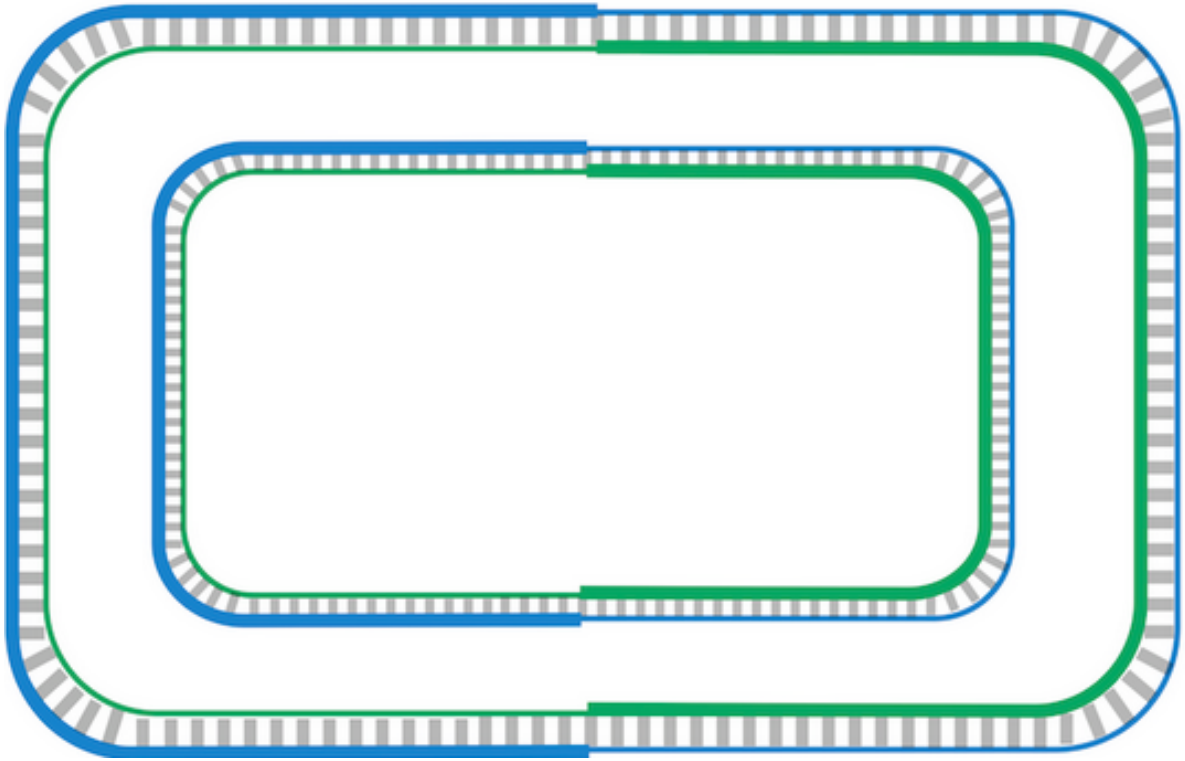
Proceso de replicación



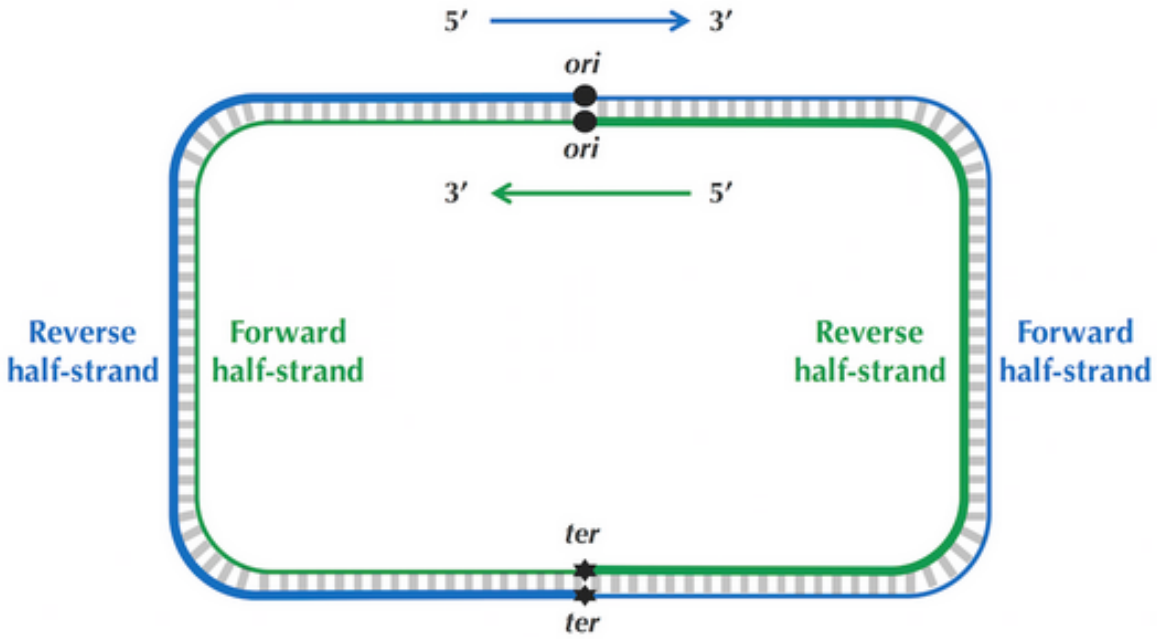
Proceso de replicación



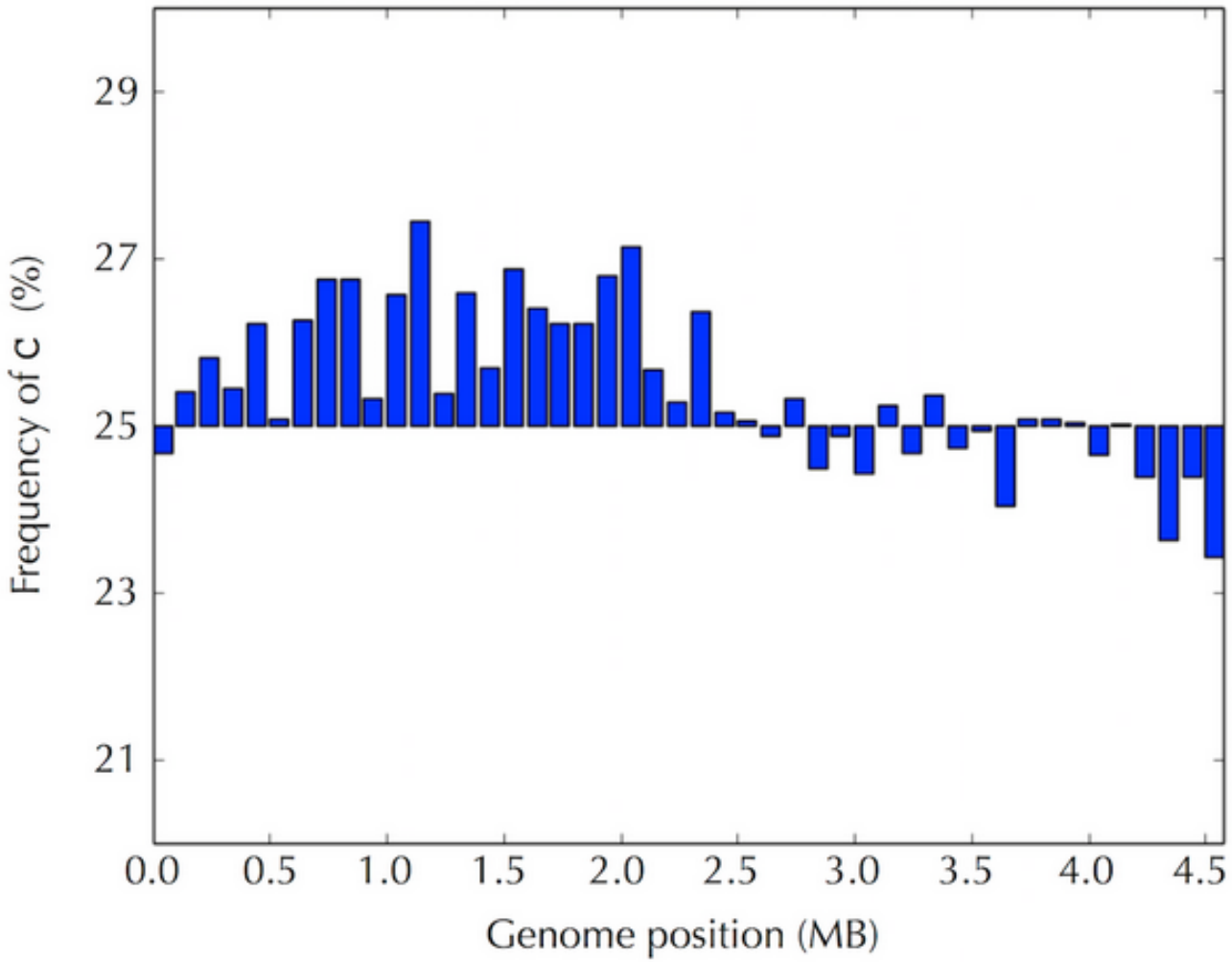
Proceso de replicación



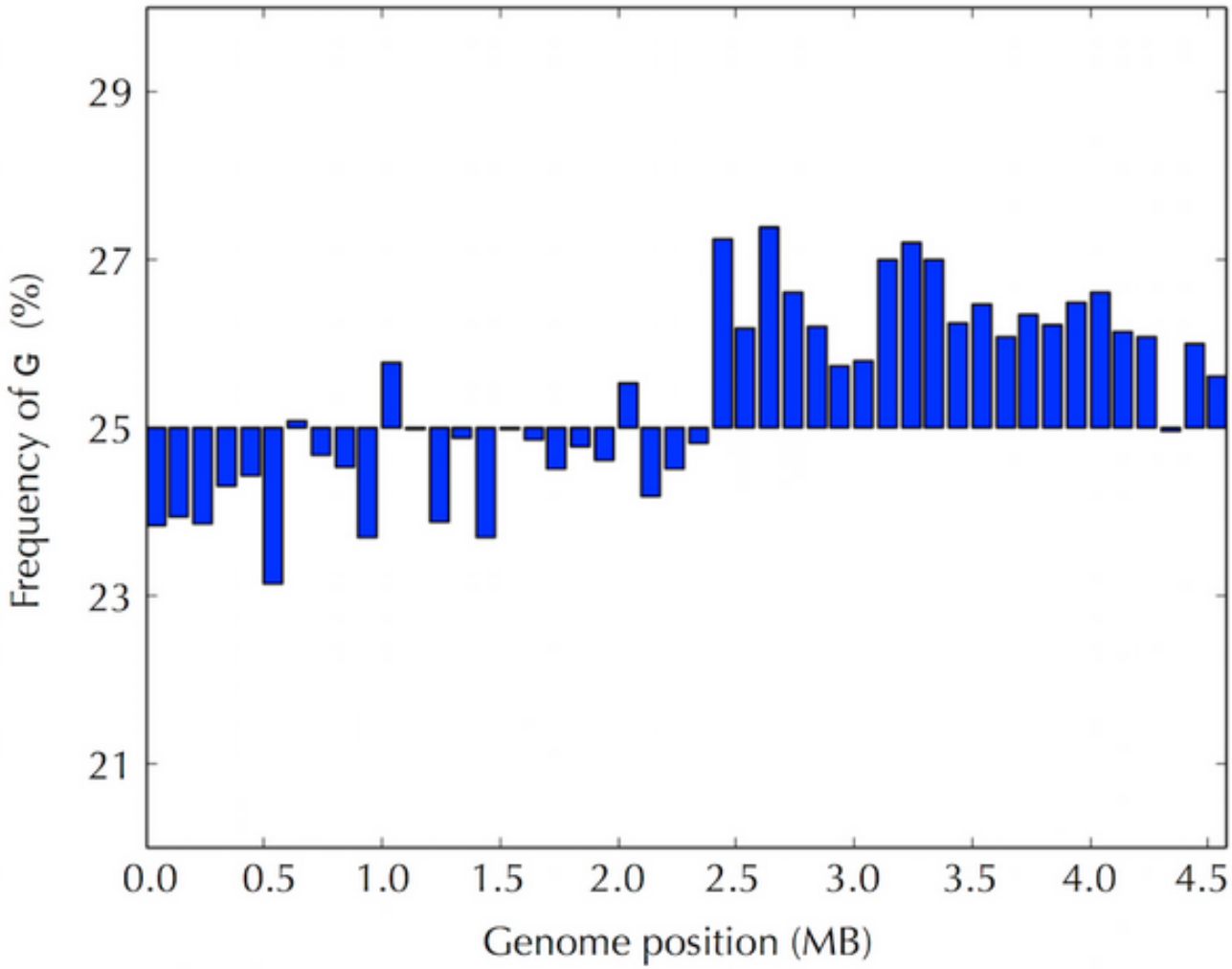
Proceso de replicación



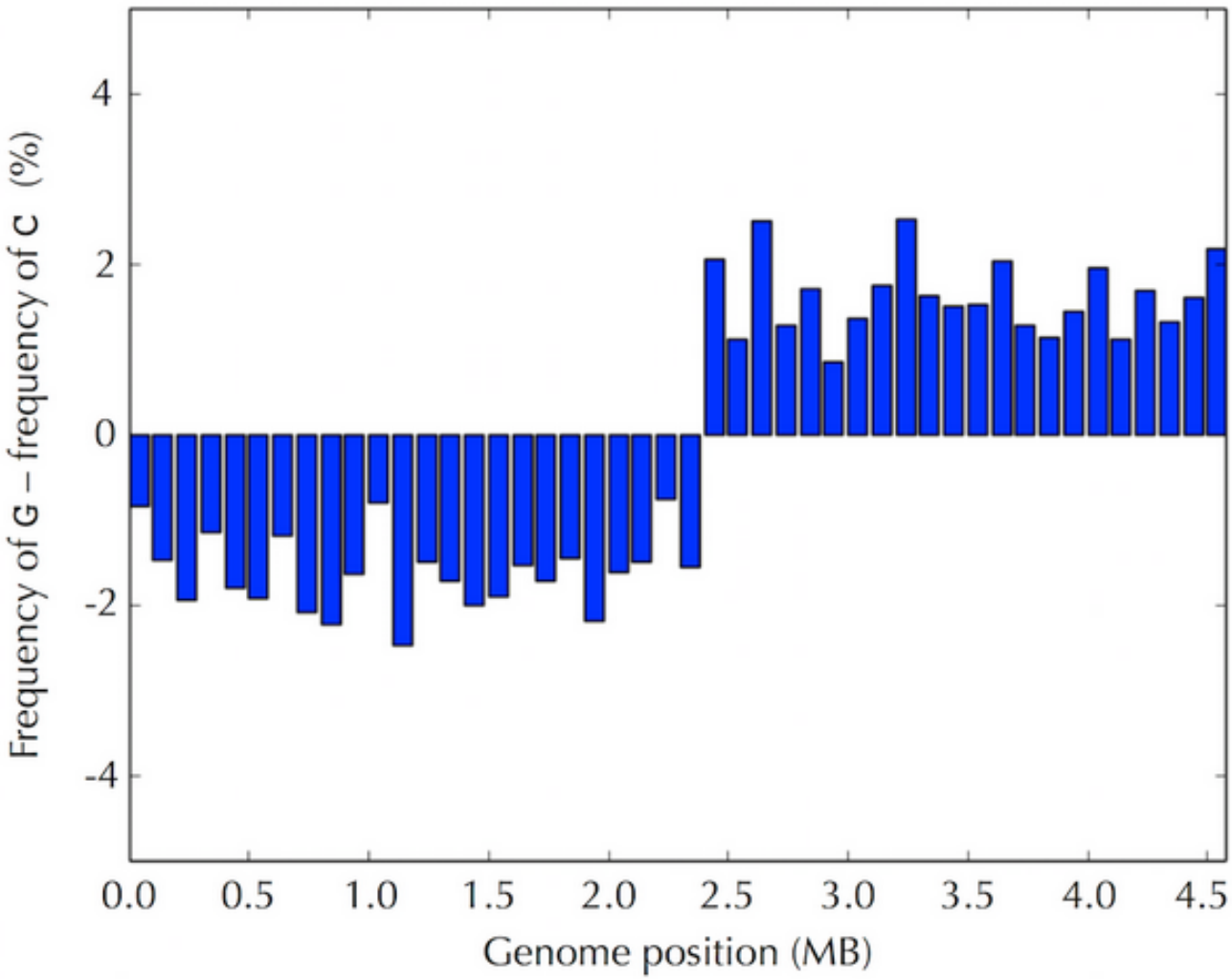
Algunas estadísticas



Algunas estadísticas



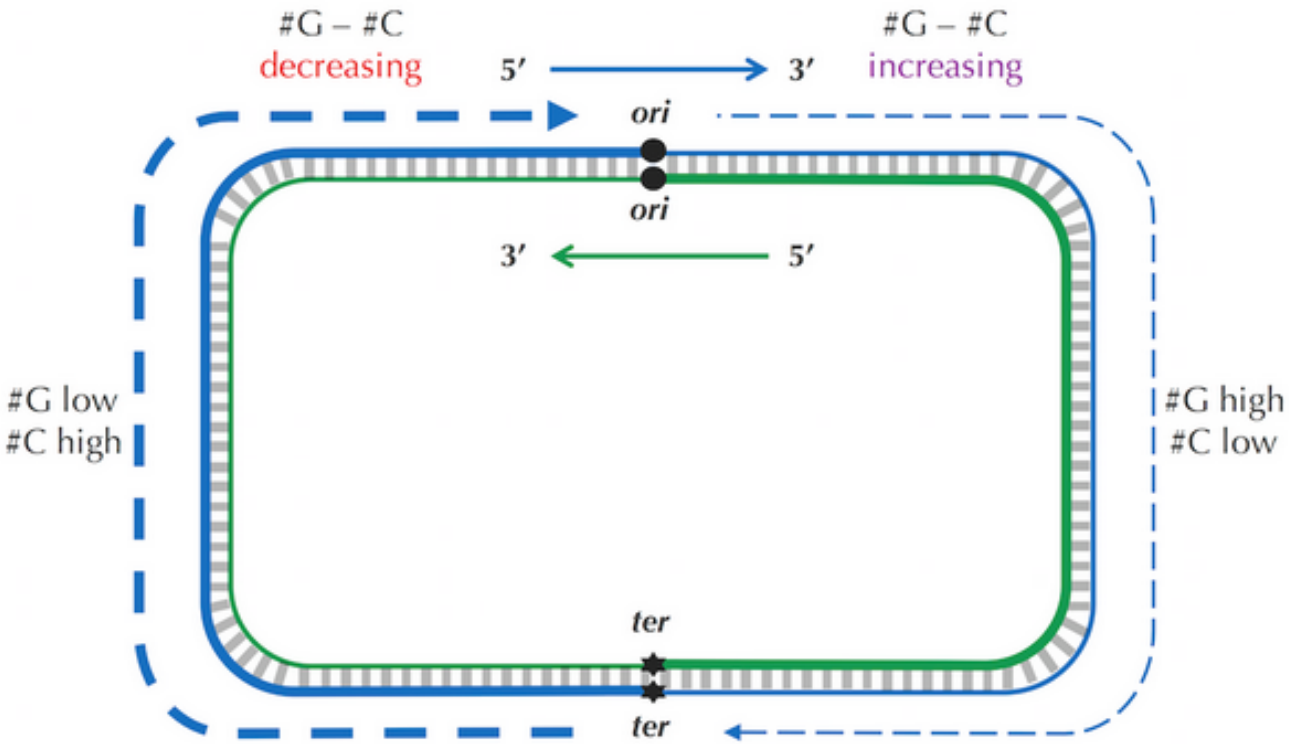
Algunas estadísticas



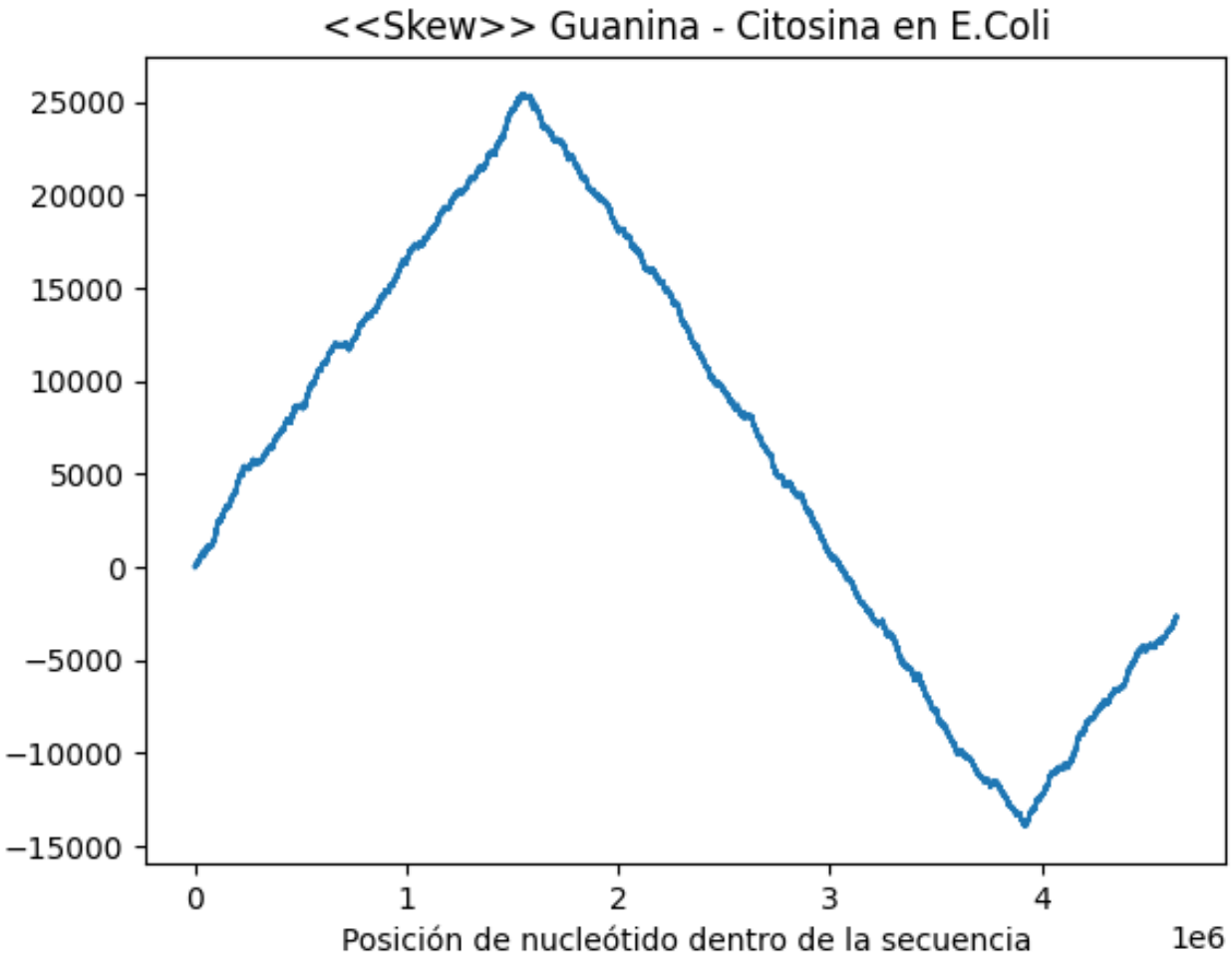
Algunas estadísticas

	#C	#G	#A	#T
Entire strand	427419	413241	491488	491363
Reverse half-strand	219518	201634	243963	246641
Forward half-strand	207901	211607	247525	244722
Difference	+11617	-9973	-3562	+1919

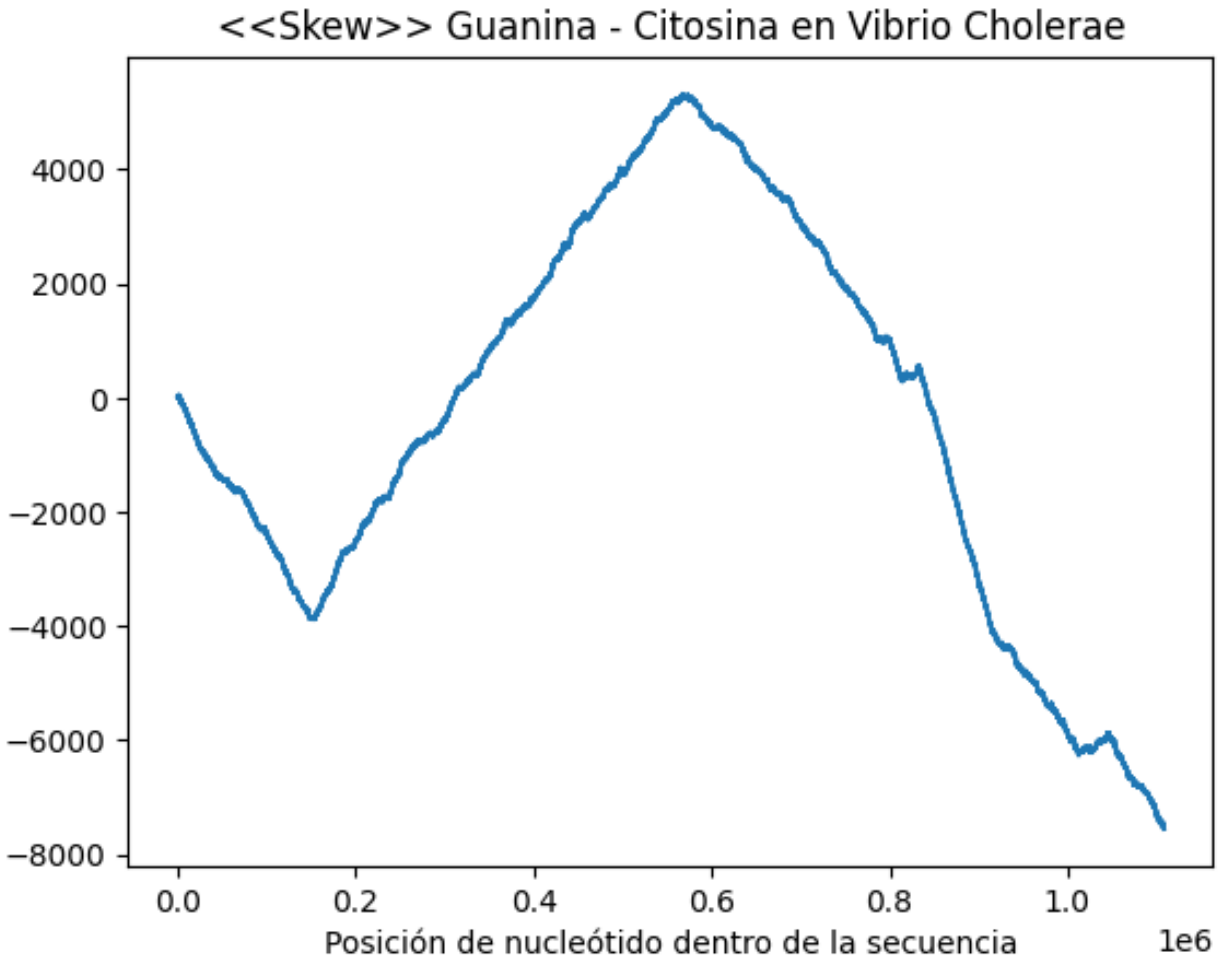
Algunas estadísticas



GC Skew de E. Coli



GC Skew de Vibrio Cholerae



Nuevo problema computacional

Dada una cadena de ADN *Genome* encontrar donde el diagrama GC Skew obtiene su valor mínimo (global o local).

¿A que me refiero con mínimo global y local?

Un detalle...

aatgatgatgacgtcaaaaggatccggataaaacatggtgattgcctcgc

ataacgcggtatgaaaatggattgaagcccgggccgtggattctactcaa

ctttgtcggcttgagaaagacctgggatcctgggtattaaaaagaagatc

tatttatttagagatctgttctattgtgatctcttattaggatcgcactg

ccctgtggataacaaggatccggcttttaagatcaacaacctggaaagga

tcattaactgtgaatgatcggtgatcctggaccgtataagctgggatcag

aatgaggggttatacacaactcaaaaactgaacaacagttgttctttgga

taactaccggttgatccaagcttcctgacagagttatccacagtagatcg

cacgatctgtatacttatttgagtaaattaacccacgatcccagccattc

ttctgccggatcttccggaatgtcgtgatcaagaatgttgatcttcagtg

Un detalle...

atcaATGATCAACgtaagcttctaagcATGATCAAGgtgctcacacagtttatccacaac

ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca

cggaaagATGATCAAGagaggatgatttcttggccatatcgcaatgaatacttgtgactt

gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt

acgaaagCATGATCATggctgttgttctgtttatcttgttttgactgagacttgttagga

tagacggttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat

tgataatgaatttacatgcttccgcgacgatttacctCTTGATCATcgatccgattgaag

atcttcaattgttaattctcttgcctcgactcatagccatgatgagctCTTGATCATgtt

tccttaaccctctatttttacggaagaATGATCAAGctgctgctCTTGATCATcgtttc

Otro problema

Dada una cadena *Genome*, patrón *Pattern* y $d \in \mathbb{N}$ encontrar todas las posiciones de aquellas observaciones que aparezcan como subcadenas en *Genome* tales tengan, a lo más, d diferencias con *Pattern*.

Nuevo problema computacional

Encontrar todos los k-meros más frequentes con a lo más d diferencias o mutaciones en una cadena. Como input necesitamos una cadena texto y $k, d \in \mathbb{N}$.

Approximate Pattern Count

```
ApproximatePatternCount(Text, Pattern, d)
    count ← 0
    for i ← 0 to |Text| - |Pattern|
        Pattern' ← Text(i , |Pattern|)
        if HammingDistance(Pattern, Pattern') ≤ d
            count ← count + 1
    return count
```

FrequentWordsWithMismatches

```
FrequentWordsWithMismatches(Text, k, d)
    Patterns ← an array of strings of length 0
    freqMap ← empty map
    n ← |Text|
    for i ← 0 to n - k
        Pattern ← Text(i, k)
        neighborhood ← Neighbors(Pattern, d)
        for j ← 0 to |neighborhood| - 1
            neighbor ← neighborhood[j]
            if freqMap[neighbor] doesn't exist
                freqMap[neighbor] ← 1
            else
                freqMap[neighbor] ← freqMap[neighbor] + 1
    m ← MaxMap(freqMap)
    for every key Pattern in freqMap
        if freqMap[Pattern] = m
            append Pattern to Patterns
    return Patterns
```

Nuevo problema computacional

Encontrar todos los k-meros más frecuentes (con diferencias y complementos inversos) en una cadena.

En este caso, el conteo es sobre todos los k-meros con a lo más d diferencias y todos los complementos inversos con a lo más d diferencias.

Origen de replicación de la E. Coli.

aatgatgatgacgtcaaaaggatccggataaaacatggtgattgcctcgc

ataacgcggtatgaaaatggattgaagcccgggccgtggattctactcaa

ctttgtcggcttgagaaagacctgggatcctgggtattaaaaagaagatc

tatttatttagagatctgttctattgtgatctcttattaggatcgcactg

cccTGTGGATAAcaaggatccggcttttaagatcaacaacctggaaagga

tcattaactgtgaatgatcggtgatcctggaccgtataagctgggatcag

aatgaggggTTATACACAactcaaaaactgaacaacagttgttcTTTGGA

TAActaccggttgatccaagcttcctgacagagTTATCCACAgtagatcg

cacgatctgtatacttatttgagtaaattaacccacgatcccagccattc

ttctgccggatcttccggaatgtcgtgatcaagaatgttgatcttcagtg

Complicaciones