

BIOINFORMATICS

SESSION 5. PRACTICE

2023-10-09

Iron imbalance and the iron responsive element

Make a Session5 directory

http://166.104.118.163:20002/



Files

Running

Clusters

Select items to perform actions on them.

<input type="checkbox"/> 0	▼	📁 / tutor
		📁 ..
<input type="checkbox"/>		📁 Session2
<input type="checkbox"/>		📁 Session3
<input type="checkbox"/>		📁 Session4

Replace Character(s) in String

stringModule.ipynb

```
1  import string
2
3  testDNA = "ATTTTATTTTATTTTA"
4  t2u = testDNA.maketrans("T", "U")
5  testRNA = testDNA.translate(t2u)
6  print ( testRNA )
7
```

AUUUUUUUUUUUUUUUUUUUU

Continued from stringModule.ipynb

```
1 testRNA2 = testDNA.replace('T', 'U')
2 print ( testRNA2 )
3
4 testRNA3 = testDNA.replace('T', 'U', 4)
5 print ( testRNA3 )
6
```

↑
count

```
AUUUUUAUUUUUAUUUUA
AUUUUATTTTATTTTA
```

String comparison

strcmp.ipynb

```
1  if "a" == "a" :
2      print ( True )
3
4  else :
5      print ( False )
6
7
8  if "a" == "A" :
9      print ( True )
10
11 else :
12     print ( False )
13
14
15 if "1" == 1 :
16     print ( True )
17
18 else :
19     print ( False )
20
```

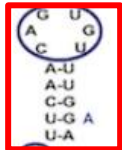
True
False
False

ire.py

- Iron response element (IRE)

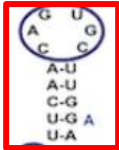
- ▣ CAGUGN motif in the loop

checking 16 nucleotide sequences
for today's practice

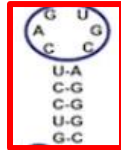


Human / mouse

Iron storage

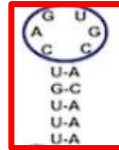


Human / mouse



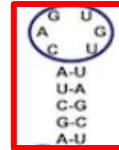
Human / mouse

Heme synthesis



Human & mouse

TCA cycle

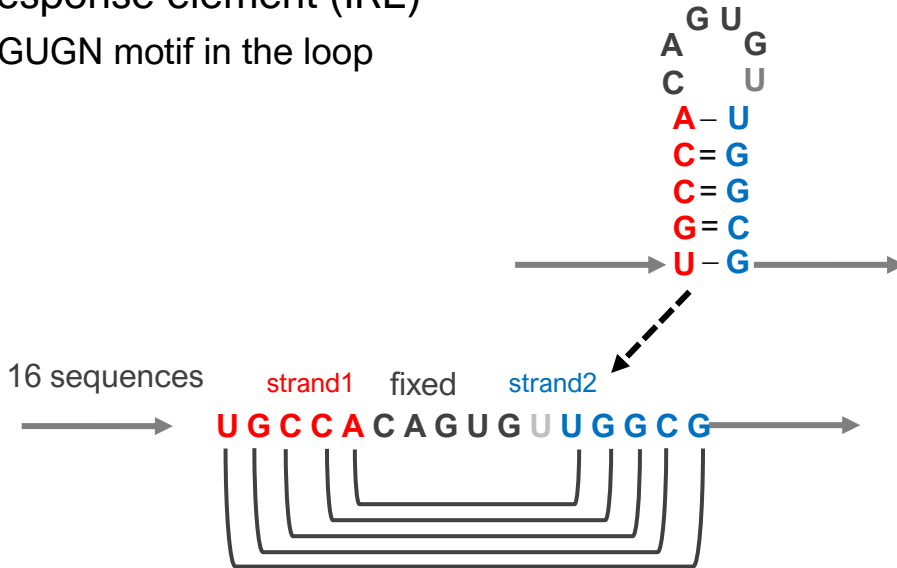


Human & mouse

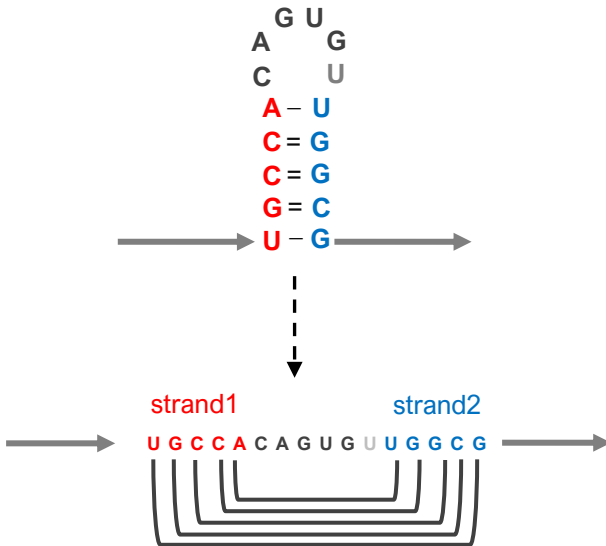
Iron export

ire.py

- Iron response element (IRE)
 - ▣ CAGUGN motif in the loop



ire.py



16 sequences

```

1 def findstem(strand1, strand2):
2     tag = True
3     for j in range(0, 5):
4         base1 = strand1[j]
5         base2 = strand2[4 - j]
6         if not pair(base1, base2):
7             tag = False
8     if tag :
9         return True
10
11
12 def pair(base1, base2):
13     if base1 == 'G' and base2 == 'C' #
14     or base1 == 'G' and base2 == 'U' #
15     or base1 == 'A' and base2 == 'U' #
16     or base1 == 'C' and base2 == 'G' #
17     or base1 == 'U' and base2 == 'A' #
18     or base1 == 'U' and base2 == 'G':
19         return True
20
21
22 seq = 'GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUUGGACGGAACCCGGCGCUCGUUCCCCA'
23
24 for i in range(0, len(seq) - 15):
25
26     test = seq[i:i + 16]
27     if test[5:10] == 'CAGUG':
28         strand1 = test[0:5]
29         strand2 = test[11:16]
30         if findstem(strand1, strand2):
31             pos = i + 1
32             print ( 'match at position', pos, ':' )
33             print ( test )
34             print ( '<----CAGUGN---->' )
35

```

match at position 23 :
 UUCAACAGUGCUUGGA
 <----CAGUGN---->

Seq= "GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUUGGACGGAACCCGGCGCUCGUUCCCCA"

ire.py result

```
match at position 23 :  
UUCAACAGUGCUUGGA  
<----CAGUGN---->
```

ire2.ipynb

match at position 23 :
UUCAACAGUGCUUGGA
((((CAGUGN))))

```
1 def findstem(strand1, strand2):
2     leftPar = ''
3     rightPar = ''
4     pairNum = 0
5     for j in range(0, 5):
6         base1 = strand1[j] # -->
7         base2 = strand2[4 - j] # <--
8         if not pair(base1, base2):
9             leftPar += '.' # -->
10            rightPar = '.' + rightPar # <--
11        else:
12            leftPar += '('
13            rightPar = ')' + rightPar
14            pairNum += 1
15    return leftPar, rightPar, pairNum
16
17 def pair(base1, base2):
18     if base1 == 'G' and base2 == 'C' #:
19     or base1 == 'G' and base2 == 'U' #:
20     or base1 == 'A' and base2 == 'U' #:
21     or base1 == 'C' and base2 == 'G' #:
22     or base1 == 'U' and base2 == 'A' #:
23     or base1 == 'U' and base2 == 'G': # G-U Wobble
24         return True # Yes, two nts base-pair
25     else:
26         return False # No, X base-pair
27
28
29 seq = 'GAGAGCAGUGGGGUUUUCCUGCUUCAACAGUGCUUGGACGGAAACCCGGCGUCGUUCCCCA'
30
31 for i in range(0, len(seq)-15):
32     test = seq[i:i+16] # test sequence of 16nt length
33     if test[5:10] == 'CAGUG':
34         strand1 = test[0:5]
35         strand2 = test[11:16]
36         leftPar, rightPar, pairNum = findstem(strand1, strand2)
37         if pairNum >= 4:
38             pos = i + 1 # i is a index of the (test) position
39             print ( 'match at position', pos, ':' )
40             print ( test )
41             print ( leftPar + 'CAGUGN' + rightPar )
42
```

Exercise 1

- There are many ways to solve a problem in programming. Consider the construction in 'ire2.py' with a number of operations with substring functions, such as:

```
if test[5:10] == 'CAGUG':
```

However, we could instead make use of a **regular expression**:

```
if re.search('( ) ( ) ( )', test)
```

In this expression, we may capture not only **a loop sequence as group(2)**, but also **strand1 and strand2 variables as group(1) and group(3)**, respectively.

[Modify 'ire2.py' to use this type of regular expression](#)

Exercise 1 answer

```
1 import re
2
3 def findstem(strand1, strand2):
4     leftPar = ''
5     rightPar = ''
6     pairNum = 0
7     for j in range(0, 5):
8         base1 = strand1[j] # -->
9         base2 = strand2[4 - j] # <--
10        if not pair(base1, base2):
11            leftPar += '.' # -->
12            rightPar = '.' + rightPar # <--
13        else:
14            leftPar += '('
15            rightPar = ')' + rightPar
16            pairNum += 1
17    return leftPar, rightPar, pairNum
18
19 def pair(base1, base2):
20    if base1 == 'G' and base2 == 'C' ¶
21    or base1 == 'G' and base2 == 'U' ¶
22    or base1 == 'A' and base2 == 'U' ¶
23    or base1 == 'C' and base2 == 'G' ¶
24    or base1 == 'U' and base2 == 'A' ¶
25    or base1 == 'U' and base2 == 'G': # G-U Wobble
26        return True # Yes, two nts base-pair
27    else:
28        return False # No, X base-pair
29
30
31 seq = 'GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUJGGACGGAACCCGGCGCUCGUUCCCCA'
32
33 for i in range(0, len(seq)-15):
34     test = seq[i:i+16] # test sequence of 16nt length
35     grouping = re.search('(.{5})(CAGUG).(.{5})', test)
36     if grouping:
37         strand1 = grouping.group(1) #(.{5}): 5 nucleotides
38         loop = grouping.group(2) # (CAGUG.) CAGUG and 1 any nucleotide
39         strand2 = grouping.group(3) #(.{5}): 5 nucleotides
40         leftPar, rightPar, pairNum = findstem(strand1, strand2)
41         if pairNum >= 4:
42             pos = i + 1 # i is a index of the (test) position
43             print ( 'match at position', pos, ':' )
44             print ( test )
45             print ( leftPar + loop + rightPar )
```

Assignment

- Modify your ire.ipynb to search the complete iron response elements (IREs) in the 'refseq_human.txt' using regular expressions.
- Note that sequences in the 'refseq_human.txt' is DNA sequences ("T" in 'refseq_human.txt" should be replaced by "U")
- * 오른쪽 그림과 같이 28nt(16nt 아님!) IRE구조를 가진 gene id, sequence를 refseq_human.txt에서 regular expression (re.search)를 이용하여 찾아보기.
- * !주의! refseq_human.txt는 'T'로 되어있으니 'U'로 변환해야함.
- * 과제 제출 기한: 10/15 Sunday 23:59 @ LMS
- * 작성한 코드와 해당 코드의 결과를 캡처한 뒤 워드에 첨부(코드만 굵어와서 붙여넣지 말기), 코드에 대한 설명 간략히 작성
워드 파일명은 n주차_학번_이름 형식으로 제출(e.g. 5주차_2023123456_김현우)

