PROJEKT G 12. 12.2018

TERMIN ROZLICZENIA 31.12.2018

BIOALGORYTMIKA: WYSZUKIWANIE WZORCÓW

PYTHON: WYRAŻENIE REGULARNE

https://pythonforbiologists.com/regular-expressions/

https://www.python-course.eu/python3_re.php

https://docs.python.org/3/library/re.html

Numer strony 141

7: Regular expressions

The importance of patterns in biology

re.py

The common theme among all these problems is that they involve searching for a **fixed** set of characters. But there are many problems that we want to solve that require more flexible patterns. For example:

- Given a DNA sequence, what's the length of the poly-A tail?
- Given a gene accession name, extract the part between the third character and the underscore
- Given a protein sequence, determine if it contains this highly-redundant domain motif

Because these types of problems crop up in so many different fields, there's a standard set of tools in Python[1] for dealing with them: *regular expressions*. Regular expressions[2] are a topic that might not be covered in a general-purpose programming book, but because they're so useful in biology, we're going to devote the whole of this chapter to looking at them.

Konstrukcja wzorca do poszukiwań:

re_0.py

re_1.py

```
if re.search(r'GAATA', DNA) or re.search(r'GATTA', DNA) :
    print(' mamy')
```

Wykorzystanie wyrażeń regularnych:

```
if re.search( r'GA(A|T)TA',  DNA) :
    print('mamy')
```

Może być albo A albo T w danej pozycji

```
if re.search( r'GA(A|T|C|G)TA',  DNA) :
    print('mamy')
```

Może być dowolna z liter A, T, C, G w danej pozycji

```
if re.search( r'GA [ATCG]TA',  DNA) :
    print('mamy')
```

Może być dowolna z liter A, T, C, G w danej pozycji

```
if re.search( r'GA.TA',  DNA) :
    print('mamy')
```

Może być dowolny znak w danej pozycji

```
if re.search( r'GA[^A]TA',  DNA) :
    print('mamy')
```

Każdy inny znak niż A w danej pozycji

ZNAKI SPECJALNEGO ZNACZENIA

znak **?** / (grupa znaków) **?**
opcjonalna jest dany znak (grupa znaków) lub go/jej nie ma

Wzorzec	Szukamy:
GAT?C	GATC GAC
GA(AAT)?C	GAAATC GAC

znak **+** /(grupa znaków) **+**
musi być, ale może występować dowolną ilość razy

GAT+C	GATC GATTC GATTTC …..
GA(AT)+C	GAATC GAATATC GAATATATATC ….

znak ***** /(grupa znaków) *****
może być, i może występować dowolną ilość razy

GAT*C	GAC GATC GATTC GATTTC …..
GA(AT)*C	GAC GAATC GAATATC …..

znak **{k}** / (grupa znaków) **{k}**
musi wystąpić daną ilość razy.,
{k,l} wystąpień od k do l.

GAT{2,4}C	GATTC GATTTC GATTTTC
GA(AT){2}C	GAATATC

^ i wzorzec – wyszukiwanie łańcuchów o zadanym początku

^GAT	GAT…….

$ za wzorcem – wyszukiwanie łańcuchów o zadanym końcu

GAT$	……..GAT

- Ten sam wzorzec może być reprezentowany na wiele sposobów
  - GG[AT]CC,
  - (GGACC|GGTCC)
  - (GGA|GGT)CC
  - G{2}[AT]C{2}

- Wyrażenia można łączyć na wiele sposobów

^ATG[ATGC]{30,1000}A{5,10}$

Start to ATG

Od 30 do 1000 dowolnych nukleotydów

Zakończonych 5 do 10 A

Przykład : Jakich sekwencji szukamy jeśli:

GA([ATGC]{3})AC([ATGC]{2})AC

GA([ATGC]{3})AC([ATGC]{2})AC

GA .. .. .. AC .. .. AC

Trzy nukleo-tydy

Dwa nukleo-tydy

Opis wyszukania

```
for dna in DNA:
   m = re.search(r"G[ATGC]{3}A", dna )
   if m :
      print(dna, ' jest ',  m.group())
   else:
      print(dna, ' brak ')
```

W każdej linii szukamy sekwencji
5 nukleotydów o początku G ,
następnie 3 dowolne znaki ,
a na końcu A: G___A

Wyrażenia_regularne_I.py

```
for dna in DNA:
   m = re.match(r"G[ATGC]{3}A", dna )
   if m :
      print(dna, ' jest ',  m.group(), mspan())
   else:
      print(dna, ' brak ')
```

Wyrażenia_funkcje.py

```
for dna in DNA:
   m = re.findall(r"G[ATGC]{3}A", dna )
   if m :
      print(dna, ' jest ',  m)
   else:
      print(dna, ' brak ')
```

```
dna = "ATGACGTACGTACGACTGACGTACCGACG"

# w zmiennej  m   przechowujemy wynik poszukiwań
m = re.search(r"GA([ATGC]{3})AC([ATGC]{2})AC", dna)

print("calkowita zgodność: " + m.group())
print("zgodnosc pierwszej cześci: " + m.group(1))
print("zgodność w drugiej częsci wzorca " + m.group(2))
```

Wyrażenia_regularne_II.py

- mamy dostęp do wydzielonych nawiasami grup.
- mamy informacje o lokalizacji

```
# w zmiennej  m   przechowujemy wynik poszukiwań
m = re.search(r"GA([ATGC]{3})AC([ATGC]{2})AC", dna)
print("calkowita zgodność: " + m.group(), ' start: ', m.start(), ' koniec: ', m.end()  )
print("zgodnosc pierwszej cześci: " + m.group(1), ' start: ', m.start(1), ' koniec: ', m.end(1) )
print("zgodnosc w drugiej częsci wzorca " + m.group(2), ' start: ', m.start(2), ' koniec: ', m.end(2))
```

Pozostałe funkcje modułu re :

- match Match a regular expression pattern to the beginning of a string.
- fullmatch Match a regular expression pattern to all of a string.
- **search** Search a string for the presence of a pattern.
- sub Substitute occurrences of a pattern found in a string.
- subn Same as sub, but also return the number of substitutions made.
- split Split a string by the occurrences of a pattern.
- findall Find all occurrences of a pattern in a string.
- finditer Return an iterator yielding a match object for each match.
- escape Backslash all non-alphanumerics in a string.

Przykład: nazwy dostępu do genów

{xkn59438, yhdck2, eihd39d9, chdsye847, hedle3455, xjhd53e, 45da, de37dp}

Podaj wzorzec , który wybiera z powyższej listy tylko te nazwy, które

zawierają liczbę 5
zawierają literę d lub e
zawierają obie litery d i e i to w tym porządku
zawierają litery d i e ,w tym porządku i tylko jedna litera pomiędzy nimi
zawierają litery d i e w dowolnym porządku
zaczynają się od x lub y

PROJEKT G

Dana jest sekwencja nukleotydów:

CGGGGCTATGCAACTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGACGTACGTACGATGCAACTCCAAAGCGGACAAA
GGATGCAACTGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGATGCAACTCGACGTACGTACGAGCTGGTTCTACCTG
AATTGGTCTAAAAAGATTATAATGTCGGTCCATGCAACTT
CTGCTGTACAACTGAGATCATGCTGCATGCAACTTTCAAC
TACATGATCTTTTGATGCAACTTGGATGAGGGAATGATGC
ATGACGTACGTACGACTGACGTACCGACGCATGCAACTTC

Wyszukać w niej (z opisem) wszystkie wystąpienia

1) zlepków nukleotydów GC
2) zawierające zlepki G…C lub C…G przy czym pomiędzy nimi jest jeden dowolny nukleotyd
3) zawierające dwa zlepki GC i CG przy czym pomiędzy nimi jest dowolna sekwencja nie dłuższa niż 6
4) kończą się na zlepku C…C przy czym pomiędzy nimi jest dowolna liczba wystąpień A