BIOINFORMATICS
SESSION 11. PRACTICE

2023-11-13

A slimy molecule

Basic Shell Commands – xShell 을 통해 서버 접속

```
$ cd [User_Folder]
$ mkdir session11
$ cd session11
```

Basic Shell Commands

```
$ cp -r /home/biguser/tutor/session11/sysModule_example .

$ cd sysModule_example/
```

Sys arguments

```
$ vi sysargv.py
```

```
1 import sys
2
3 for i in sys.argv:
4     print(i)
5
6 infile1= open(sys.argv[1])
7 infile2= open(sys.argv[2])
8 infile3= open(sys.argv[3])
9
10 def readfile(filename):
11     for line in filename:
12         print(line)
13
14 readfile(infile1)
15 readfile(infile2)
16 readfile(infile3)
```

Sys arguments

```
$ python sysargv.py file1.txt file2.txt file3.txt
```

```
[biguser@R440 sysModule_example]$ python sysargv.py file1.txt file2.txt file3.txt
sysargv.py
file1.txt
file2.txt
file3.txt
Today is Tuesday

2023-11-13

Bioinformatics
```

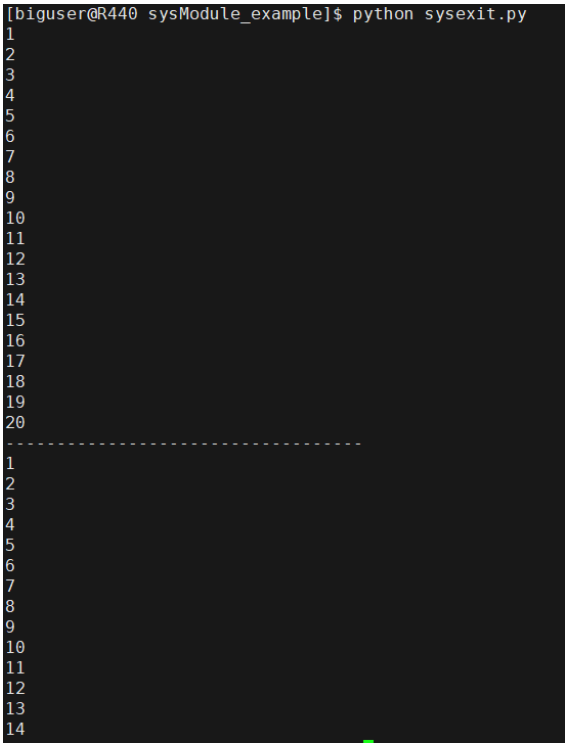
Sys exit

```
$ vi sysexit.py
```

```
 1 import sys
 2
 3 for i in range(1,21):
 4     print(i)
 5
 6 print('-----------------------------------')
 7
 8 for i in range(1,21):
 9     if i==15:
10         sys.exit()
11     else:
12         print(i)
13
```

Sys exit

```
$ python sysexit.py
```



Counting specific amino acid in sliding windows Code 13.1 -- pts.py ($ vi pts.py)

```
 3 import re
 4 import sys
 5
 6 # Basic parameters used
 7
 8 wid  = 100  # size of sliding window
 9 step = 1    # size of step to move sliding window
10
11 # check if argument to the script is there.
12
13 if len(sys.argv) > 1:
14     file = sys.argv[1]
15 else:
16     sys.exit('File in FASTA sequence format is to be used as argument to the script')
17
18 # read the sequence from the input file
19
20 seq = ''
21 id = ''
22
23 for line in open(file):
24     line = line.rstrip()
25
26         # in the identifier line all is captured
27         # in the variable 'id' except for
28         # the > character
29
30     match = re.search('>(.*)', line)
31     if match:
32         id = match.group(1)
33     else:
34         seq = seq + line
```

```
36 # Now analyze the sequence in $seq
37
38 print('Position\tProline\tThreonine\tSerine')
39
40 for i in range(0, len(seq) - wid+1, step):
41
42     test = seq[i:i + wid]
43
44     # Count proline, threonine and serine
45
46     count_p = float(test.count('P')) / wid
47     count_t = float(test.count('T')) / wid
48     count_s = float(test.count('S')) / wid
49     pos = i + 1 + wid / 2
50     print(pos, '\t', count_p, '\t', count_t, '\t', count_s)
```

Code 13.1
pts.py

```
$ cp /home/biguser/tutor/session11/muc6.fa .

$ python pts.py muc6.fa

$ python pts.py muc6.fa > pts.out

$ less pts.out
```

```
[biguser@R440 session11]$ python pts.py muc6.fa
Position        Proline Threonine       Serine
51      0.05    0.08    0.11
52      0.05    0.08    0.11
53      0.05    0.08    0.11
54      0.05    0.08    0.11
55      0.05    0.08    0.12
56      0.05    0.08    0.12
57      0.05    0.08    0.12
58      0.05    0.09    0.12
59      0.05    0.09    0.12
60      0.05    0.09    0.12
61      0.05    0.09    0.12
62      0.05    0.09    0.12
63      0.05    0.09    0.12
64      0.05    0.09    0.12
65      0.05    0.09    0.13
66      0.05    0.09    0.13
67      0.05    0.09    0.12
68      0.05    0.09    0.12
69      0.05    0.09    0.12
70      0.05    0.09    0.12
71      0.05    0.09    0.12
72      0.05    0.09    0.12
73      0.05    0.08    0.13
74      0.05    0.08    0.12
75      0.06    0.08    0.12
76      0.06    0.07    0.12
77      0.06    0.08    0.11
78      0.05    0.08    0.12
79      0.05    0.08    0.12
80      0.05    0.08    0.12
81      0.05    0.08    0.12
82      0.05    0.08    0.12
83      0.05    0.08    0.12
84      0.05    0.09    0.12
85      0.06    0.09    0.12
86      0.06    0.09    0.11
87      0.05    0.09    0.11
88      0.05    0.09    0.11
```

Visualization of PTS landscape with R

```
$ cp /home/biguser/tutor/session11/pts.r
$ vi pts.r
```

```
# read information from output from Python script
data <- read.table("pts.out", sep = "\t", header = TRUE)

# make an empty plot
plot(0, type = "n", xlim = c(0, 2500), ylim = c(0,
    0.45), main = "PTS domain", xlab = "Position", ylab = "Score")

# draw lines for Proline, Serine and Threonine data

lines(data$Position, data$Proline, col = "blue", lwd = 2)
lines(data$Position, data$Serine, col = "green", lwd = 2)
lines(data$Position, data$Threonine, col = "red",
    lwd = 2)

# make a legend
legend(50, 0.4, c("Thr", "Ser", "Pro"), col = c("red",
    "green", "blue"), lwd = 2)

# add a line indicating the 40% / 5% cutoff

len <- length(data$Position) # number of lines in the file

for (i in (1:len)) {
    if (((data$Serine[i] + data$Threonine[i]) > 0.4) && (data$Proline[i] >
        0.05)) {
        points(i, 0, col = "darkgrey")
    }
}
dev.off()
```

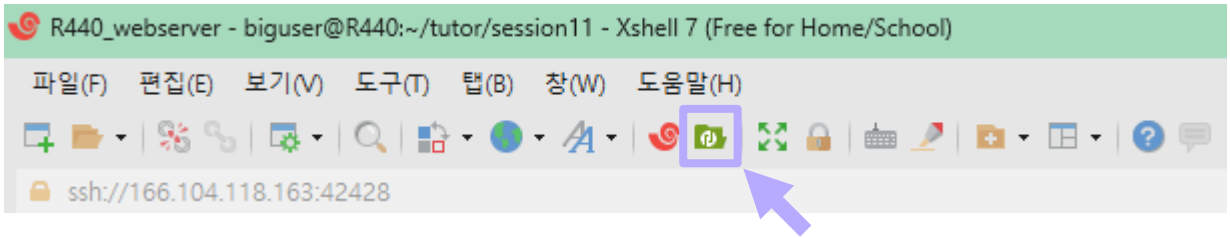
`type`
what type of plot should be drawn. Possible types are

- `"p"` for points,
- `"l"` for lines,
- `"b"` for both,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both 'overplotted',
- `"h"` for 'histogram' like (or 'high-density') vertical lines,
- `"s"` for stair steps,
- `"S"` for other steps, see 'Details' below,
- `"n"` for no plotting.

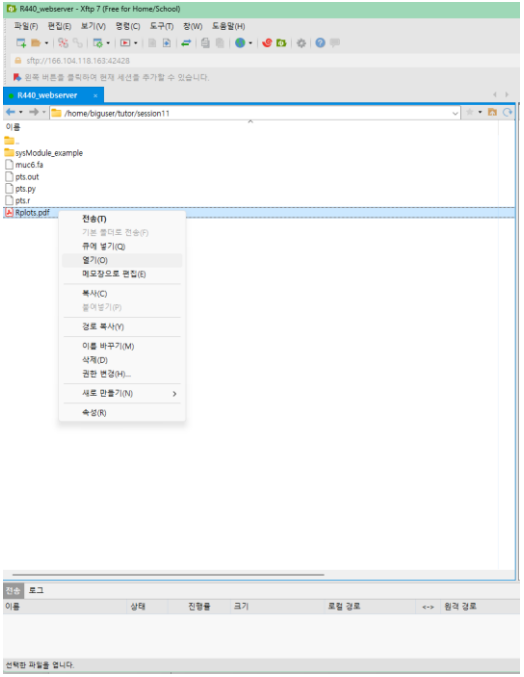
Visualization of PTS landscape with R

```
$ Rscript pts.r
```

```
[biguser@R440 session11]$ ll
total 104
-rw-r--r-- 1 biguser biguser  2527 Nov  9 14:55 muc6.fa
-rw-rw-r-- 1 biguser biguser 59282 Nov  9 15:36 pts.out
-rw-r--r-- 1 biguser biguser  1097 Nov  9 15:15 pts.py
-rw-r--r-- 1 biguser biguser   894 Nov  9 14:55 pts.r
-rw-rw-r-- 1 biguser biguser 29601 Nov  9 15:36 Rplots.pdf
drwxr-xr-x 2 biguser biguser   144 Nov  9 15:06 sysModule_example
```

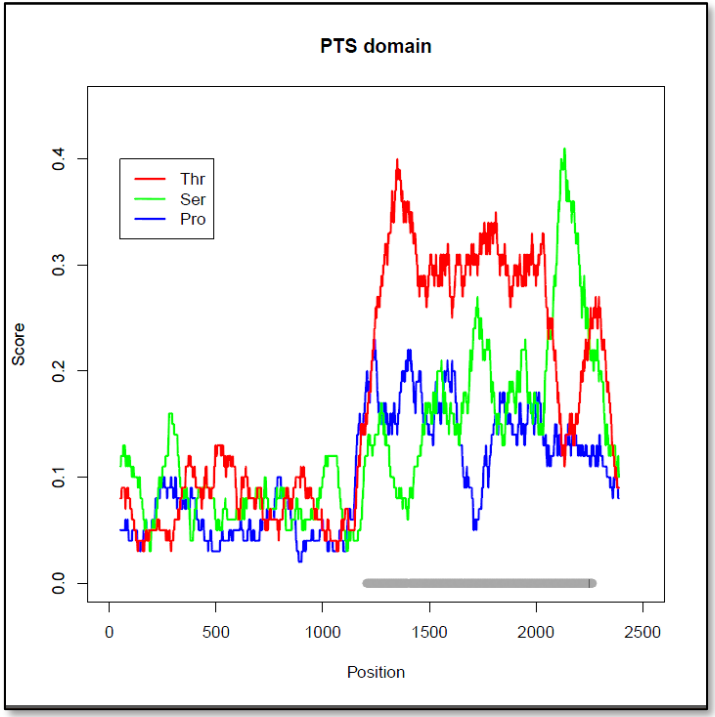


Visualization of PTS landscape with R



Right click and click open on “Rplot.pdf”

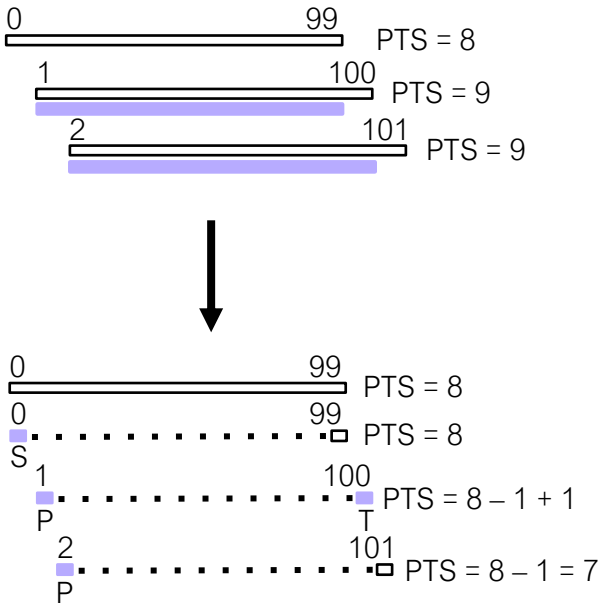
Visualization of PTS landscape with R



Exercise 13.1- Efficient programming

- The counting of amino acids in Code 13.1 is not optimal as we are analyzing overlapping windows of the mucin sequence and are therefore examining the same amino acid positions several times. Modify Code 13.1 to avoid this situation.

Exercise 13.1 - Example of efficient programming




```
print("Position", '\t', "Proline", '\t', "Threonine", '\t', "Serine")

test = seq[0:0+wid]
count_p = test.count('P')
count_t = test.count('T')
count_s = test.count('S')
frac_p  = float(count_p) / wid
frac_s  = float(count_s) / wid
frac_t  = float(count_t) / wid
pos     = 1
print(pos + wid/2, '\t', frac_p, '\t', frac_t, '\t', frac_s)
for i in range(1, len(seq) - wid+1, step):
    minus = seq[i-1]
    plus = seq[i + wid-1]

    minus_p = minus.count('P')
    minus_s = minus.count('S')
    minus_t = minus.count('T')

    plus_p = plus.count('P')
    plus_s = plus.count('S')
    plus_t = plus.count('T')

    count_p = count_p - minus_p + plus_p
    count_s = count_s - minus_s + plus_s
    count_t = count_t - minus_t + plus_t

    frac_p = float(count_p) / wid
    frac_s = float(count_s) / wid
    frac_t = float(count_t) / wid
    pos = i + 1
    print(pos + wid/2, '\t', frac_p, '\t', frac_t, '\t', frac_s)
```

Exercise 13.2

- In the code pts.py, we count amino acids using the count operator. Modify the script to show that the counting could also be carried out using either of “re.findall()” and “re.subn()”:

re.findall()

```
>>> test="PPPTTTSSS"
>>> import re
>>> p_count=re.findall("P", test)
>>> p_count
['P', 'P', 'P']
```

re.subn()

returns the replaced string and the number of replacement

```
>>> test="PPPTTTSSS"
>>> import re
>>> p_count= re.subn("(P)", "", test)
>>> p_count
('TTTSSS', 3)
```

Exercise 13.1

```
for i in range(0, len(seq) - wid, step):
    test = seq[i:i + wid]

    # re.findall()
    count_p= re.findall("P", test);count_p = len(count_p); count_p= float(count_p)/ wid
    count_t= re.findall("T", test);count_t = len(count_t); count_t= float(count_t)/ wid
    count_s= re.findall("S", test);count_s = len(count_s); count_s= float(count_s)/ wid

    pos = i + 1 + wid/2
    print(pos, '\t', count_p, '\t', count_t, '\t', count_s)

    # re.subn()
    count_p = re.subn('(P)', '', test) ; count_p = float(count_p[1]) / wid
    count_t = re.subn('(T)', '', test) ; count_t = float(count_t[1]) / wid
    count_s = re.subn('(S)', '', test) ; count_s = float(count_s[1]) / wid
    pos = i + 1 + wid/2
    print(pos, '\t', count_p, '\t', count_t, '\t', count_s)
```

Assignment

- The mucin sequence analyzed in pts.py (**muc6.fa**) contains repetitive sequences. Construct a Python script to examine **every possible word of size four** (i.e. every sequence of four consecutive amino acids) and **count the number of times that word occurs** in the mucin sequence. What are the **top5 most common four-letter word** and **how many times do they occur** in MUC6 in muc6.fa?

- muc6.fa 를 활용하여 가능한 모든 4aa 길이의 sequence가 muc6에서 반복되는 횟수를 확인하고 가장 빈번하게 나타나는 repetitive sequence top5개 sequence와 해당 횟수를 print하는 코드를 구현하세요.

- 힌트 -> dictionary 활용 + sorted()

- 과제 제출 기한: 11/19 Sunday 23:59 @ LMS
- 작성한 코드와 해당 코드의 결과를 캡처한 뒤 워드에 첨부(코드만 긁어와서 붙여넣지 말기), 코드에 대한 설명 간략히 작성
  워드 파일명은 n주차_학번_이름 형식으로 제출(e.g. 11주차_2023123456_김현우)