Session 11. PRACTICE

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
```

```
import sys

for i in sys.argv:
    print i

infile1= open(sys.argv[1])
infile2= open(sys.argv[2])
infile3= open(sys.argv[3])

def readfile(filename):
    for line in filename:
        print line

readfile(infile1)
readfile(infile2)
readfile(infile3)
```

Sys arguments

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

```
(py27) [biguser@R440 sysModule_example]$ python sysargv.py file1.txt file2.txt file3.txt
sysargv.py
file1.txt
file2.txt
file3.txt
Today is Wednesday

2022-11-15

Bioinformatics
```

Sys exit

```
$ vi sysexit.py
```

```
import sys

for i in range(1,21):
    print i

print '-----------------------------------'

for i in range(1,21):
    if i==15:
        sys.exit()
    else:
        print i
```

Sys exit

```
$ python sysexit.py



$ cd ..
```

```
[kyungtae@biglab-master sysModule_example]$ python sysexit.py
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
-----------------------------------
1
2
3
4
5
6
7
8
9
10
11
12
13
14
[kyungtae@biglab-master sysModule_example]$
```

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

```
#!/usr/bin/python

import re
import sys

# Basic parameters used

wid  = 100  # size of sliding window
step = 1    # size of step to move sliding window

# check if argument to the script is there.

if len(sys.argv) > 1:
    file = sys.argv[1]
else:
    exit('File in FASTA sequence format is to be used as argument to the script'
         )

# read the sequence from the input file

seq = ''
id = ''

for line in open(file):
    line = line.rstrip()

        # in the identifier line all is captured
        # in the variable 'id' except for
        # the > character

    match = re.search('>(.*)', line)
    if match:
        id = match.group(1)
    else:
        seq = seq + line
```

```
# Now analyze the sequence in $seq

print 'Position\tProline\tThreonine\tSerine'

for i in range(0, len(seq) - wid+1, step):

    test = seq[i:i + wid]

    # Count proline, threonine and serine

    count_p = float(test.count('P')) / wid
    count_t = float(test.count('T')) / wid
    count_s = float(test.count('S')) / wid
    pos = i + 1 + wid / 2
    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
```

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	

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`

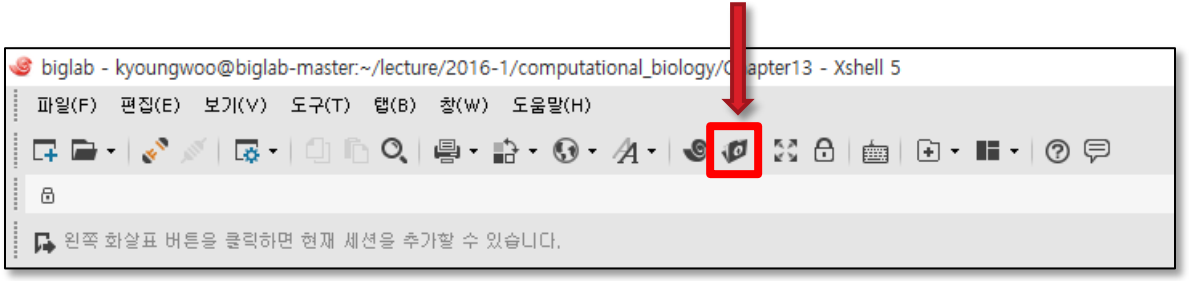
what type of plot should be drawn. Possible types are

- `"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.

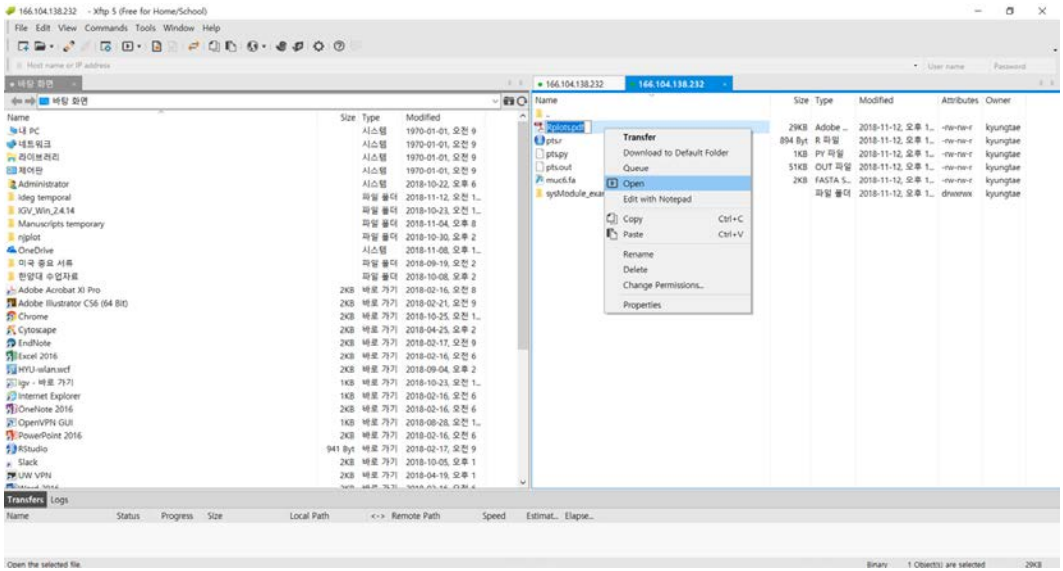
Visualization of PTS landscape with R

```
$ R
➢ source(“pts.r”)
Or
$ Rscript pts.r
```

```
[biguser@biglab-master session10]$ ll
total 88K
-rw-rw-r-- 1 biguser biguser 52K Nov 13 00:01 pts.out
-rw-rw-r-- 1 biguser biguser 894 Nov 13 00:01 pts.r
-rw-rw-r-- 1 biguser biguser 29K Nov 13 00:01 Rplots.pdf
```

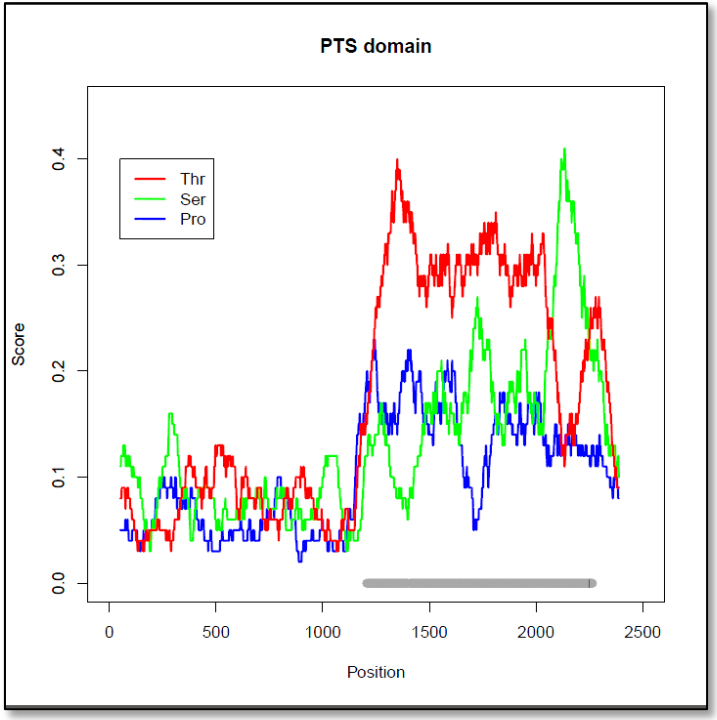


Visualization of PTS landscape with R



Mouse right click and click open from “Rplot.pdf”

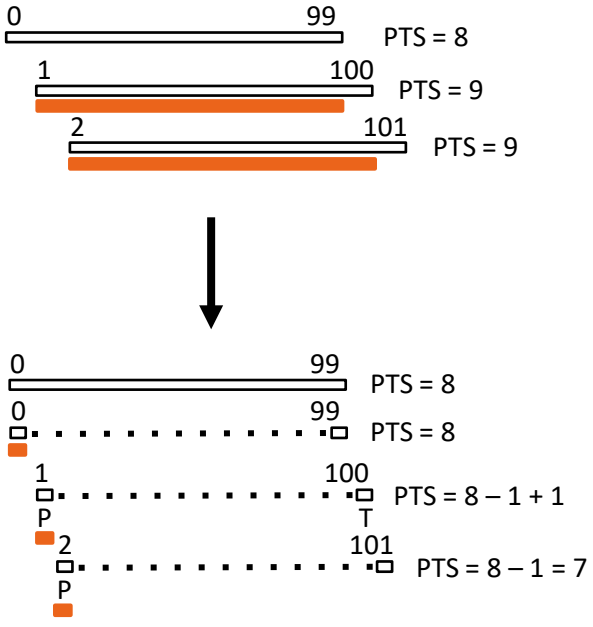
Visualization of PTS landscape with R



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.

Example of efficient programming




```
import re
import sys

wid  = 100
step = 1


if len(sys.argv) > 1:
    file = sys.argv[1]
else:
    exit('File in FASTA sequence format is to be used as argument to the script')


seq = ''
id = ''

for line in open(file):
    line = line.rstrip()
    match = re.search('>(.*)', line)
    if match:
        id = match.group(1)
    else:
        seq = seq + line

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 13.1

- In Code 13.1 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()

```
>>> 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]
    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
```

```
for i in range(0, len(seq) - wid, step):
    test = seq[i:i + wid]
    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 Code 13.1 (**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 the mucin MUC6?
- 힌트 -> dictionary 활용 + sorted

- 제출기한: 11/21 (월요일) 오후 6시까지