Genome Sciences 373
Genome Informatics

Quiz Section 4

April 21, 2015

Topics today

- Questions about homework
- Smith-Waterman algorithm: local alignment
- Reading files in python
- Functions in python

Smith-Waterman alignment

- Local alignment means:
  - We don’t have to end at the bottom right
  - We don’t have to end at the top left

- Best alignment may only be a single pair of nucleotides!

S-W: what to check when finished

- All cells with positive numbers should have arrows **pointing in**
  - (how did I get here?)

- …but not necessarily **pointing out**

- **Calculate** the alignment score by hand and double-check your work

Let’s align two sequences:

`CGTTA & GACGT`

Note: they don’t have to be the same length!

substitution matrix

	A	**C**	**G**	**T**
A	4	-2	0	-2
C		4	-2	0
G			4	-2
T				4

gap penalty -3, linear

		C	G	T	T	A
	0					
G						
A						
C						
G						
T						

		C	G	T	T	A
	0	0	0	0	0	0
G	0					
A	0					
C	0					
G	0					
T	0					

		C	**G**	**T**	**T**	**A**
	0	0	0	0	0	0
G	0	0				
A	0					
C	0					
G	0					
T	0					

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4			
A	0					
C	0					
G	0					
T	0					

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	0	0	1	2	0	4
C	0	4	1	0	0	1
G	0	1	8	5	2	0
T	0	0	5	12	9	6

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	0	0	1	2	0	4
C	0	4	1	0	0	1
G	0	1	8	5	2	0
T	0	0	5	12	9	6

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	0	0	1	2	0	4
C	0	4	1	0	0	1
G	0	1	8	5	2	0
T	0	0	5	**12**	9	6

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	0	0	1	2	0	4
C	0	4	1	0	0	1
G	0	1	**8**	5	2	0
T	0	0	5	**12**	9	6

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	0	0	1	2	0	4
C	0	**4**	1	0	0	1
G	0	1	**8**	5	2	0
T	0	0	5	**12**	9	6

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	**0**	0	1	2	0	4
C	0	**4**	1	0	0	1
G	0	1	**8**	5	2	0
T	0	0	5	**12**	9	6

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	X	0	1	2	0	4
C	0	**4**	1	0	0	1
G	0	1	**8**	5	2	0
T	0	0	5	**12**	9	6

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	X	0	1	2	0	4
C	0	4	1	0	0	1
G	0	1	8	5	2	0
T	0	0	5	12	9	6

Best local alignment is:

CGT
CGT

S-W: what to check when finished

- All cells with positive numbers should have arrows **pointing in**
  - (how did I get here?)

- …but not necessarily **pointing out**

- **Calculate** the alignment score by hand and double-check your work

		C	G	T	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	X	0	1	2	0	4
C	0	**4**	1	0	0	1
G	0	1	**8**	5	2	0
T	0	0	5	**12**	9	6

gap penalty -3, linear

substitution matrix

	A	C	G	T
A	4	-2	0	-2
C		4	-2	0
G			4	-2
T				4

```
Best local
alignment is:
   CGT
   CGT

Calculate the score
```

		C	G	C	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	0	0	1	2	0	4
C	0	4	1	5	2	1
G	0	1	8	5	3	0
T	0	0	5	8	9	6

Here we make a small change to one of the sequences

		C	G	C	T	A
	0	0	0	0	0	0
G	0	0	4	1	0	0
A	0	0	1	2	0	4
C	0	4	1	5	2	1
G	0	1	8	5	3	0
T	0	0	5	8	9	6

Note that our score drops and then goes up again!

Practice problem

align **TGCATT** and **GGCA** using Smith-Waterman local alignment

Gap = -3

	A	C	T	G
A	3	−2	−2	−1
C	−2	3	−1	−2
T	−2	−1	3	−2
G	−1	−2	−2	3

Practice problem

align **TGCATT** and **GGCA** using Smith-Waterman local alignment

Gap = -3

	A	C	T	G
A	3	−2	−2	−1
C	−2	3	−1	−2
T	−2	−1	3	−2
G	−1	−2	−2	3

Answer:

GCA

GCA

with score = 9

Reading files: several options

Example code to read just one line:

```
my_filename = sys.argv[1]
my_open_file= open(my_filename, “r”)

# read just the first line
my_first_line = my_open_file.readline()

# now I can read another line
my_second_line = my_open_file.readline()
my_third_line = my_open_file.readline()
```

Reading files: several options

Example code to read it all at once:

```
my_filename = sys.argv[1]
my_open_file= open(my_filename, “r”)

# read all of my file at once.
# note: if your file is really big (like, say, >1Gb)
# then you do NOT want to do this!
my_entire_file = my_open_file.read()

# split it into a list of strings
my_lines = my_entire_file.split(“\n”)

for my_line in my_lines:
   do_something()
```

Reading files: several options

Example code to read all lines, one at a time:

```
my_filename = sys.argv[1]
my_open_file= open(my_filename, "r")

num_lines = 0
for my_line in my_open_file:
   my_line = my_line.strip()  # chop off the "\n" at the end
   do_something()
   num_lines += 1
print "I found %d lines" % num_lines

# alternative way
for my_line in my_open_file.readlines():
   my_line = my_line.strip()  # chop off the "\n" at the end
   do_something()
```

Functions in Python: a brief overview

You've already seen several functions in python:

int(*argument***)**	convert *argument* to an integer, return the integer
float(*argument)*	convert *argument* to a float, return the float
len(*argument*)	calculate the length of *argument*, return the length

Functions in Python: a brief overview

Functions are:

reusable pieces of code, that
take zero or more **arguments**,
perform some **actions**, and
return one or more values

Functions in Python: a brief overview

Functions are:

- **reusable** pieces of code, that
- take zero or more **arguments**,
- perform some **actions**, and
- **return** one or more values

conceptually

function "**sum**"
- takes arguments a, b
- adds a and b
- returns sum

Functions in Python: a brief overview

Functions are:

reusable pieces of code, that
take zero or more **arguments**,
perform some **actions**, and
return one or more values

conceptually

function "**sum**"
    takes arguments a, b
    adds a and b
    returns sum

in python...

```
def sum(a, b):
    total = a + b
    return total

# later in the program
my_sum = add(2, 5)
# my_sum is now 7
```

Functions in Python: a brief overview

Functions are:

reusable pieces of code, that take zero or more **arguments**, perform some **actions**, and **return** one or more values

in python…

stuff that happens in here is invisible outside of the function

```
def sum(a, b):
    total = a + b
    return total

# later in the program
my_sum = add(2, 5)
print total   # this won't work!
```

In-class example:

Write a function to calculate the factorial of an integer