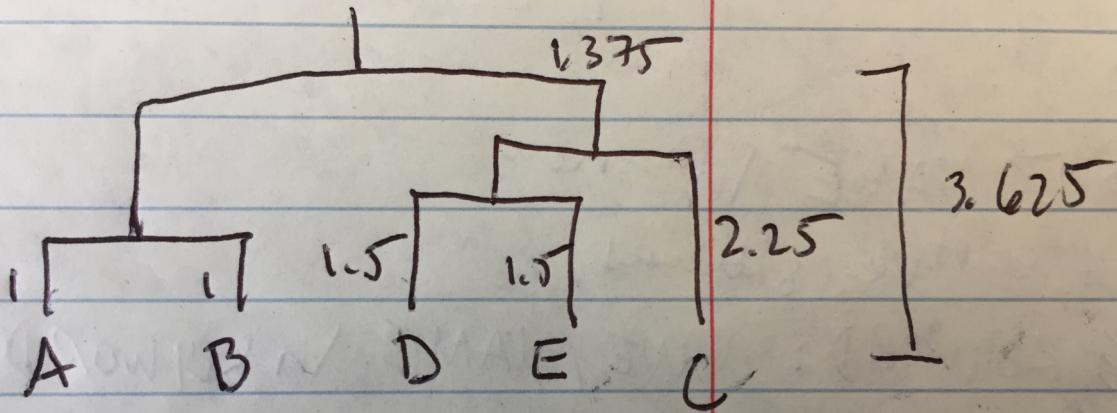


Genome Informatics Quiz section week 3

April 12, 2018

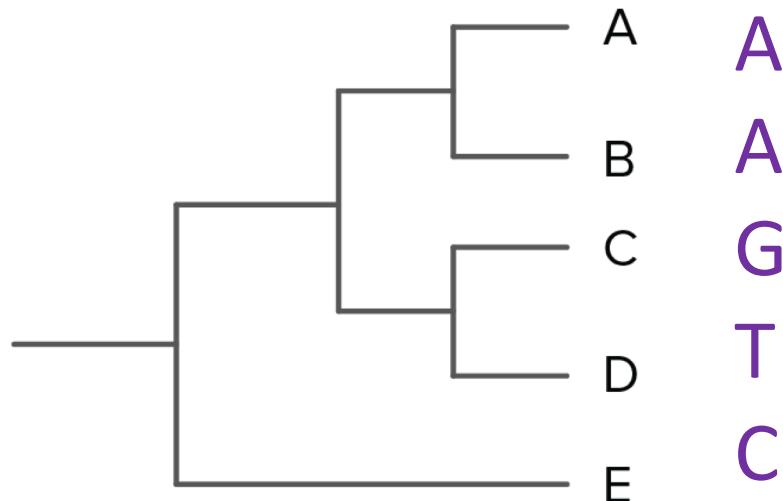
UPGMA

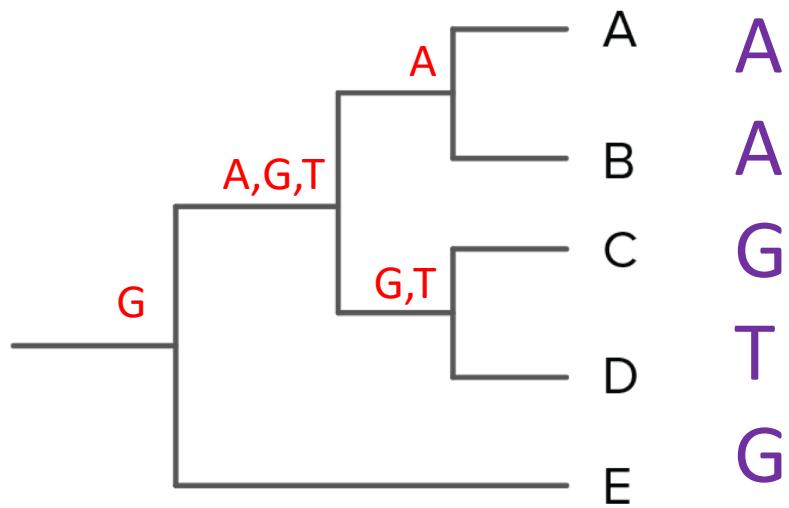
	A	B	C	D	E
A	0				
B	2	0			
C	6	5	0		
D	10	9	4	0	
E	9	8	5	3	0



Fitch algorithm: What are we doing?

- The *small*/parsimony problem
- Analyzing a *single* tree
 - Min changes required (parsimony score)
 - Parsimonious assignment of internal node traits





Parsimony score: 2

A few missed questions:

- Scoping – (if you don't know what it is, ignore me for a moment, we'll discuss again later) – in Python, there is no block-level scope, variables declared in loops and conditionals are in the same scope as surrounding code
- String find all – short answer, no built-in function for this, but you could easily build your own after we discuss loops!

for loops

for loops let you repeatedly apply the same code to all elements in a list

```
>>> l = [1, 2, 3]
>>> print l[0]
>>> print l[1]
>>> print l[2]
```

for loops

for loops let you repeatedly apply the same code to all elements in a list

```
>>> l = [1, 2, 3]
>>> for item in l:
...     print item
1
2
3
```

Also works for strings

```
DNA = 'actg'  
for i in DNA:  
    print i
```

a
c
t
g

for loop

- Allows you to perform an operation on each element in a list.

The diagram illustrates the structure of a for loop. On the left, the keyword 'for' is followed by a placeholder '<target>' and 'in' followed by another placeholder '<object>'. A vertical brace on the right groups three lines of code: '<statement>', '<statement>', and '...'. To the right of the brace, the text 'Repeated block of code' is written. Three callout boxes point from the text 'Variable name available inside loop' (pointing to the placeholder '<target>'), the text 'Must be defined in advance' (pointing to the placeholder '<object>'), and the text 'Repeated block of code' (pointing to the grouped statements). A vertical bar on the far left is labeled 'Must be indented'.

```
for <target> in <object>:  
    <statement>  
    <statement>  
    ...  
    <statement>
```

Variable name available inside loop

Must be defined in advance

Repeated block of code

Must be indented

For loop examples:

```
DNA = 'AGTCGA'
```

```
base 0 is A
```

```
base 1 is G
```

```
base 2 is T
```

```
base 3 is C
```

```
base 4 is G
```

```
base 5 is A
```

Solution

```
>>> index = 0
>>> for base in DNA:
...     print "base", index, "is", base
...     index = index + 1
...
...
```

For loop examples:

```
num_list = [2, 3, 4, 5]
```

sum of all numbers?

For loop examples:

```
num_list = [2, 3, 4, 5]
```

sum of all numbers?

```
sum = 0
for v in num_list:
    sum = sum + v
print 'The sum is:', sum
```

The range() function returns a list of integers covering a specified range

```
range( [start,] stop [,step] )
```

[optional arguments],
default to 0 and 1

```
range(5)
```

```
[0, 1, 2, 3, 4]
```

```
range(2,8)
```

```
[2, 3, 4, 5, 6, 7]
```

```
>>> range(-1, 2)
```

```
[-1, 0, 1]
```

```
>>> range(0, 8, 2)
```

```
[0, 2, 4, 6]
```

```
>>> range(0, 8, 3)
```

```
[0, 3, 6]
```

```
>>> range(6, 0, -1)
```

```
[6, 5, 4, 3, 2, 1]
```

Range can be useful in a for loop

```
num_list = [0,1,2,3,4,5,6]
# print only every other number:
for i in range(0, 7, 2):
    print num_list[i]
```

Nested loops

```
nuucs = 'ACGT'  
for i in nuucs:  
    for j in nuucs:  
        print i,j
```

Terminating a loop

```
break # jumps out of the most  
      # internal loop
```

```
>>> for integer in range(0,3):  
...     if (integer == 1):  
...         break  
...     print integer  
...
```

Output?

Terminating a loop

```
continue # skips to end of current  
          # iteration of the loop
```

```
>>> for integer in range(0,3):  
...     if (integer == 1):  
...         continue  
...     print integer  
... 
```

Sample problem

Write a program `add-arguments.py` that reads any number of integers from the command line and prints the cumulative total for each successive argument.

```
> python add-arguments.py 1 2 3
```

```
1
```

```
3
```

```
6
```

```
> python add-arguments.py 1 4 -1
```

```
1
```

```
5
```

```
4
```

