Genome 373: Genome Informatics

Quiz section #2

April 5, 2018

Today (into tomorrow)

- Review alignment
- Review p-values and multiple testing
- Debugging
- More python

Local alignment example

Substitution matrix:

	A	C	G	T
A	10	-5	0	-5
C	-5	10	-5	0
G	0	-5	10	-5
T	-5	0	-5	10

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \\ 0 \end{cases}$$

(corresponds to start of alignment)

		G	A	G	T	A
	0	?				
A						
G						
T						
T						
A						

Linear gap penalty d = -4

Local alignment example

Substitution matrix:

	A	C	G	T
A	10	-5	0	-5
C	-5	10	-5	0
G	0	-5	10	-5
T	-5	0	-5	10

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \\ 0 \end{cases}$$

(corresponds to start of alignment)

		G	A	G	T	A
	0	0	0	0	0	0
A	0	?				
G	0					
T	0					
T	0					
A	0					

Linear gap penalty d = -4

Local alignment example

Substitution matrix:

	A	C	G	T
A	10	-5	0	-5
C	-5	10	-5	0
G	0	-5	10	-5
T	-5	0	-5	10

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \\ 0 \end{cases}$$

(corresponds to start of alignment)

		G	A	G	T	A
	0	0	0	0	0	0
A	0	0				
G	0	?				
T	0					
T	0					
A	0					

Linear gap penalty d = -4

Local alignment example

Substitution matrix:

	A	C	G	T
A	10	-5	0	-5
C	-5	10	-5	0
G	0	-5	10	-5
T	-5	0	-5	10

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \\ 0 \end{cases}$$

(corresponds to start of alignment)

		G	A	G	T	A
	0	0	0	0	0	0
A	0	0				
G	0	10				
T	0	6				
T	0	2				
A	0	?				

Linear gap penalty d = -4

Local alignment example

Substitution matrix:

	A	C	G	T
A	10	-5	0	-5
C	-5	10	-5	0
G	0	-5	10	-5
T	-5	0	-5	10

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \\ 0 \end{cases}$$

(corresponds to start of alignment)

		G	A	G	T	A
	0	0	0	0	0	0
A	0	0	10	6	2	10
G	0	10	6	20	16	12
T	0	6	6	16	30	26
T	0	2	2	12		
A	0	0	12	8	22	36

Maximum score

Local alignment example

Substitution matrix:

	A	C	G	T
A	10	-5	0	-5
C	-5	10	-5	0
G	0	-5	10	-5
T	-5	0	-5	10

GAGT-A
AGTTA

		G	A	G	T	A
	0	0	0	0	0	0
A	0	0	10	6	2	10
G	0	10	6	20	16	12
T	0	6	6	16	30	26
T	0	2	2	12	26	25
A	0	0	12	8	22	36

P-values!

- P-values tell you about expectations *under the null hypothesis*
  - they say *nothing* about the alternative hypothesis or how probable it is

- Null hypothesis: usually the boring default, devil's advocate position – what you want to see if you can disprove

P-values!

- P-values tell you about expectations *under the null hypothesis*
  - they say *nothing* about the alternative hypothesis or how probable it is

- Null hypothesis: usually the boring default, devil's advocate position – what you want to see if you can disprove

  There is *no difference* between treatment groups
  Life expectancy is *not changing* over time
  This coin is *not weighted*
  These two sequences are *unrelated*

Historic example: R.A. Fisher and the tea-tasting test

8 cups of tea, randomly chosen to either have tea poured over milk or milk poured over tea

Null hypothesis?

How surprising would her picks be if she was guessing randomly?

Null distribution: What we suppose the data might look like if the null hypothesis is true

- This could be based on a parameterized probability distribution
  - E.g. Poisson: number of successes in x tries with y% probability of success

- Or you can generate an *empirical* null based on your real data
  - E.g. Shuffle the labels of the variable you want to test

- Defining the most appropriate null distribution is a relevant and tough problem in a lot of computational biology research!

Multiple testing can be dangerous!

- http://fivethirtyeight.com/features/you-cant-trust-what-you-read-about-nutrition/
- Nutrition & lifestyle questionnaires from 54 individuals

Our shocking new study finds that ...

EATING OR DRINKING	IS LINKED TO
Raw tomatoes	Judaism
Egg rolls	Dog ownership
Energy drinks	Smoking
Potato chips	Higher score on SAT math vs. verbal
Soda	Weird rash in the past year
Shellfish	Right-handedness
Lemonade	Belief that "Crash" deserved to win best picture
Fried/breaded fish	Democratic Party affiliation
Beer	Frequent smoking
Coffee	Cat ownership
Table salt	Positive relationship with Internet service provider
Steak with fat trimmed	Lack of belief in a god
Iced tea	Belief that "Crash" didn't deserve to win best picture
Bananas	Higher score on SAT verbal vs. math
Cabbage	Innie bellybutton

SOURCE: FFQ & FIVETHIRTYEIGHT SUPPLEMENT

Multiple testing can be dangerous!

- http://fivethirtyeight.com/features/you-cant-trust-what-you-read-about-nutrition/
- Nutrition & lifestyle questionnaires from 54 individuals

Our shocking new study finds that ...

EATING OR DRINKING	IS LINKED TO	P-VALUE
Raw tomatoes	Judaism	<0.0001
Egg rolls	Dog ownership	<0.0001
Energy drinks	Smoking	<0.0001
Potato chips	Higher score on SAT math vs. verbal	0.0001
Soda	Weird rash in the past year	0.0002
Shellfish	Right-handedness	0.0002
Lemonade	Belief that "Crash" deserved to win best picture	0.0004
Fried/breaded fish	Democratic Party affiliation	0.0007
Beer	Frequent smoking	0.0013
Coffee	Cat ownership	0.0016
Table salt	Positive relationship with Internet service provider	0.0014
Steak with fat trimmed	Lack of belief in a god	0.0030
Iced tea	Belief that "Crash" didn't deserve to win best picture	0.0043
Bananas	Higher score on SAT verbal vs. math	0.0073
Cabbage	Innie bellybutton	0.0097

SOURCE: FFQ & FIVETHIRTYEIGHT SUPPLEMENT

Bonferroni correction: just divide the threshold by the total # of tests

- For 1000 tests: Use a threshold 1000x stricter
  - Does not require tests to have a particular relationship with each other
  - Ensures that the probability of rejecting a true null hypothesis is still less than your original desired p-value threshold

  - Suppose they did 1000 tests for this study (50 lifestyle qs and 200 foods)

New cutoff is 0.05/1000 .00005

Our shocking new study finds that ...

EATING OR DRINKING	IS LINKED TO	P-VALUE
Raw tomatoes	Judaism	<0.0001
Egg rolls	Dog ownership	<0.0001
Energy drinks	Smoking	<0.0001
Potato chips	Higher score on SAT math vs. verbal	0.0001
Soda	Weird rash in the past year	0.0002
Shellfish	Right-handedness	0.0002
Lemonade	Belief that "Crash" deserved to win best picture	0.0004
Fried/breaded fish	Democratic Party affiliation	0.0007
Beer	Frequent smoking	0.0013

- FYI: Sometimes this is too harsh, and *false discovery rate* corrections can be more useful

Python

Review - variables and types

- A bit more formally than last time, Python has 5 built-in data types
  - Number
  - String
  - List
  - Tuple (haven't discussed yet)
  - Dictionary

- You can find an object's type using the type function:

```
x = 5
type(5)
```

Numbers

- Subtypes of numbers are integers (`int`) and floats (`float`)
- Try:

```
4/3
4/3.
4./3
```

- Output of operations will be of the most complex type of the input (int/int = int, int/float = float)

- To explicitly change between types, use `int()` and `float()`

Operators

**	exponentiation	3**2 = 9
*	multiplication	2*4 = 8
/	division	4/2 = 2
+	addition	4 + 4 = 8
-	subtraction	4 – 2 = 2

Remember PEMDAS?

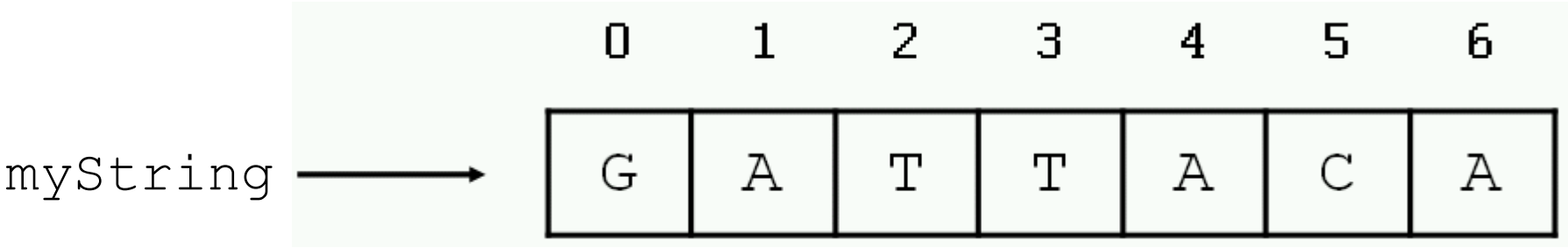
2*3**2 = ?

Strings

A series of characters starting and ending with single or double quotes

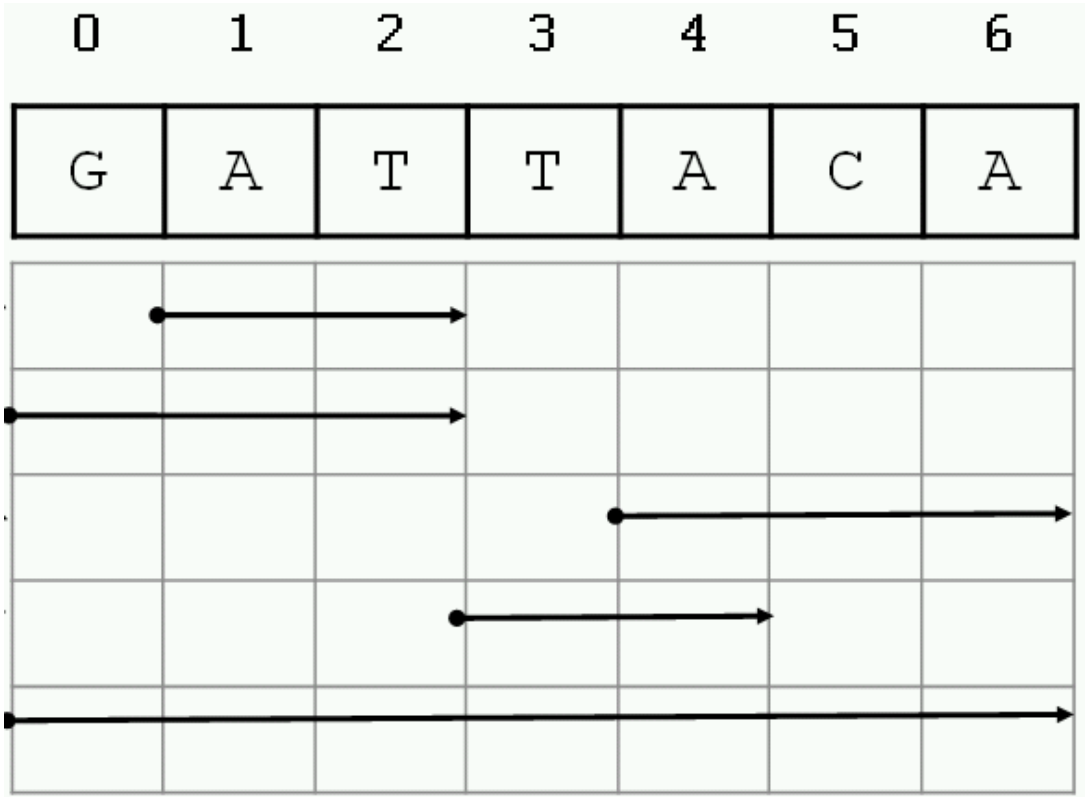
Stored as a list of characters in memory

```
>>> myString = "GATTACA"
```



Accessing substrings

```
>>> myString[1:3]
>>> myString[:3]
>>> myString[4:]
>>> myString[3:5]
>>> myString[:]
```



What about `myString[-2]`

String functionality

```
>>> len("GATTACA")

>>> "GAT" + "TACA"

>>> "A" * 10

>>> "GAT" in "GATTACA"

>>> "AGT" in "GATTACA"
```

Methods

In Python, a method is a function defined for a particular object type

The syntax is
`<object>.<method>(<parameters>)`

```
DNA = "AGT"
DNA.find("A")
0
```

String methods

```
>>> "GATTACA".find("ATT")
>>> "GATTACA".count("T")
>>> "GATTACA".lower()
>>> "gattaca".upper()
>>> "GATTACA".replace("G", "U")
>>> "GATTACA".replace("C", "U")
>>> "GATTACA".replace("AT", "**")
>>> "GATTACA".startswith("G")
>>> "GATTACA".startswith("g")
```

Strings are immutable

- String methods do not modify the string; they return a new string

```
>>> sequence = "ACGT"
>>> sequence.replace("A", "G")
'GCGT'
>>> print sequence
ACGT

>>> sequence = "ACGT"
>>> new_sequence = sequence.replace("A", "G")
>>> print new_sequence
GCGT
```

Reading input from the command line

When you type
python hannahs_program.py 2 3

python sees a list of strings:

```
[“hannahs_program.py”, “2”, “3”]
```

```
import sys # Many functions only
available via packages, you must
import them
```

Reading input from the command line

When you type

> python hannahs_program.py 2 3

python sees a list of strings and runs the first entry:

```
["hannahs_program.py", "2", "3"]
```

You can access the other parts of the list using sys.argv

Reading input from the command line

python hannahs_program.py 2 3.2

```
## Inside hannahs_program.py:
# Many functions only available via
# packages, you must import them
import sys
first_num = int(sys.argv[1])
second_num = float(sys.argv[2])
```

Sample problem

Write a program called dna2rna.py that reads a DNA sequence from the first command line argument, and then prints it as an RNA sequence. Make sure it works for both uppercase and lowercase input.

```
> python dna2rna.py  AGTCAGT
ACUCAGU
> python dna2rna.py actcagt
acucagu
> python dna2rna.py ACTCagt
ACUCagu
```

Solution

```
import sys #allow us to pull from command line

DNA = sys.argv[1]

RNA = DNA.replace("T", "U")
RNA = RNA.replace("t", "u")

print RNA


#RNA = DNA.replace("T", "U").replace("t", "u")
```

Lists

- An ordered series of objects

```
>>> list1 = ["hannah", "C", 3, 2.4]
>>> list2 = [1, 2, 3]
>>> list3 = [list1, list2]
>>> list3
[["hannah", "C", 3, 2.4], [1, 2, 3]]
```

Unlike strings, lists can be changed

```
>>> list1 = ["hannah", "C", 3, 2.4]
>>> list1[1] = "hannah”
>>> list1
["hannah", "hannah", 3, 2.4]
```

Expanding lists

```
>>> newlist = []
>>> print newlist
[]
>>> newlist.append(4)
>>> print newlist
[4]
>>> newlist.extend([4,5]
>>> print newlist
[4,4,5]
```

Handy list methods

```
L.append(x) # add x to the end of L
L.extend([x,y]) # add x and y to L
L.count(x) # count how many times x is in L
L.index(x) # give the location of x
L.remove(x) # remove first occurrence of x
L.reverse(x) # reverse order of elements of L
L.sort() # sort L
```

Tuples

- Tuples are immutable lists – you can't change them in place (like strings)
- If you want to change them, you have to assign them to a new tuple

```
>>> T = (1,2,3)
>>> T[1] = 1 # Error
>>> T = T + T
>>> T
(1,2,3,1,2,3)
```

Sample problem

Write a program that takes a list of words and prints them out in sorted order

```
> python sort_list.py hannah john george
['george', 'hannah', 'john']
```

Solution

```
import sys

iList = sys.argv[1:]

iList.sort()

print iList
```

Sample problem

Write a program that takes a DNA sequence as the first command line argument and prints the number of A's, T's, G's, and C's

```
> python dna-composition.py
ACGTGCGTTAC
2 A's
3 C's
3 G's
3 T's
```

Solution

```
import sys

DNA = sys.argv[1]

As = DNA.count('A')
Gs = DNA.count('G')
Ts = DNA.count('T')
Cs = DNA.count('C')

print "%s A's\n%s C's\n%s G's\n%s
T's\n" % (As, Cs, Gs, Ts)
```

A bit more on conditionals

```
DNA = "AGTGGT"
if (DNA.startswith("A")):
    print "Starts with A"
```

- A *block* is a group of lines of code that belong together.

```
if (<test evaluates to true>):
  <execute this block of code>
```

- In interactive mode, the ellipse indicates that you are inside a block.
- Python uses indentation to keep track of blocks.
- You can use any number of spaces to indicate blocks, but you must be consistent.
- An unindented or blank line indicates the end of a block.

Interactive on whitespace

Sample problem

- Write a program `find-base.py` that takes as input a DNA sequence and a nucleotide. The program should print where the nucleotide occurs in the sequence, or a message saying it's not there.

```
> python find-base.py A GTAGCTA
A occurs at position 3.
> python find-base.py A GTGCT
A does not occur at all.
```

Hint: `string.find('G')` returns -1 if it can't find the requested sequence.

Solution

```
import sys

base = sys.argv[1]
dna = sys.argv[2]

position = dna.find(base)

if base in dna:
print "%s occurs in position %s" %(base,
position+1)
else:
print "%s does not occur at all." % base
```