**Path 218 Handout 7**

**A. Post-Discussion Notes/Code from second discussion (Thursday 7/3/14)**

**B. Problem Set 2 (due Tuesday 7/15** @8:42pm**)**.

Problem set 2 problems are 1b,2b,2c,2d,3b,3c,3d below (1a,2a,3a1,and 3a2 were examples done in section).

Problem set philosophy for the Path 218.

0. Don't panic *(citation: Douglas Adams... "A Hitchhikers Guide to the Galaxy)*

1. The goal of this problem set is to provide practice in using variables, loops and lists in python.

2. If you fully succeed in whatever time you allot to this, wonderful (and tell us the answers)

3. If you don't fully succeed in whatever time you allot, send us what you have, pasting answers and code into a copy of this document as far as you get. If you're code isn't working, add a few comment lines explaining what you think is going wrong (you can also paste any 'error' statement after the code.

4. The problems are designed to be addressed using coding tools from first two lectures and discussion sections. But if you are familiar or want to use other *Pythonic* tools not covered in the lectures/handouts/sections, please feel free to use these (there are many solutions to any problem with Python... we're happy with any of them).

5. This homework will be due Tuesday, July 15th at 8:42pm PDT. Email your homework (as a modified version of this file) to **path218homework@gmail.com. Please title the file as follows: "YourName\_PS2.docx"**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Any comments on problem set 2?

Enter your comments here

**Problem 1. What's in a name?**

**1a.** An excercise with names. Paste the following lines into the live interpreter (remove leading spaces)

A = ['hey','I','just','met','you']

B = ['and','this','is','crazy']

C = A

A.extend(B)

What is A? ['hey','I','just','met','you', 'and','this','is','crazy']

What is C? ['hey','I','just','met','you', 'and','this','is','crazy']

C.extend(['but',"here's",'my','number'])

What is A? ['hey','I','just','met','you', 'and','this','is','crazy', 'so','call','me','maybe']

What is C? ['hey','I','just','met','you', 'and','this','is','crazy', 'so','call','me','maybe']

A = ['so','call','me','maybe']

What is A? ['so','call','me','maybe']

What is C? ['hey','I','just','met','you', 'and','this','is','crazy', 'so','call','me','maybe']

This is why we need to be careful if we want to get a copy of a list to work with independently:

A = C[:] Sets up a new list (which will have the name "A") that is formally a derivative of sequence C

that has all elements of C (essentially a copy of C that can be operated on independently)

**1b. An excercise in prediction**

**•** Make a list with **First\_Name\_For\_List =['goose','sluice','juice','kaboose']**

• Give the list a second name Second\_Name\_For\_List= First\_Name\_For\_List

• Give the list a thirs name Third\_Name\_For\_List= First\_Name\_For\_List

• Give a command to modify the underlying variable that doesn't change the associated name

First\_Name\_For\_List.append('moose')

• Did this change First\_Name\_For\_List?\_\_\_\_\_\_\_\_\_\_\_\_\_

• Did this also change Second\_Name\_For\_List? \_\_\_\_\_\_\_\_\_\_\_\_\_

• Did this also change Third\_Name\_For\_List? \_\_\_\_\_\_\_\_\_\_\_\_\_

• Try again with the second name, again a command that changes the variable in place without changing

the association between the variable and its name

Second\_Name\_For\_List.sort()

• Did this change First\_Name\_For\_List?\_\_\_\_\_\_\_\_\_\_\_\_\_

• Did this change Second\_Name\_For\_List? \_\_\_\_\_\_\_\_\_\_\_\_\_

• Did this also change Third\_Name\_For\_List? \_\_\_\_\_\_\_\_\_\_\_\_\_

• Now give a command that associates the name First\_Name\_For\_List with a new variable

First\_Name\_For\_List=['aardvark']

• Did this change First\_Name\_For\_List?\_\_\_\_\_\_\_\_\_\_\_\_\_

• Did this change Second\_Name\_For\_List? \_\_\_\_\_\_\_\_\_\_\_\_\_

• Did this also change Third\_Name\_For\_List? \_\_\_\_\_\_\_\_\_\_\_\_\_

• Now give a command that associates the name Second\_Name\_For\_List with a new variable

Second\_Name\_For\_List=sorted(Second\_Name\_For\_List)

• Did this change First\_Name\_For\_List?\_\_\_\_\_\_\_\_\_\_\_\_\_

• Did this change Second\_Name\_For\_List? \_\_\_\_\_\_\_\_\_\_\_\_\_

• Did this also change Third\_Name\_For\_List? \_\_\_\_\_\_\_\_\_\_\_\_\_

• Bonus: After A=1 and B=A, would you expect either A=A+1 or A+=1 to change B? \_\_\_ \_\_\_

**Problem 2. Where oh where could my little worm be?**

**•** File "myDogsRNA.txt" contains a list of small RNA sequence reads taken from a cell population derived from our trusty dog Fido (actually we don't have a dog and if we did, he/she wouldn 't be named fido, but these *are* real sRNA reads from a canine lymphocyte sample sequenced at the Max Planck Institute in Berlin). The sequences in myDogsRNA.txt have no barcode at the beginning (they start immediately with the small RNA sequence) but they do have a linker at the end with a sequence that starts TCGT...

**2a.** Generate a list of dog miRNAs and their representation number in myDogsRNA.txt.

The **solution** (below) is a very slight variant on our in-class example from Lecture 2. Remember that any data files you want to open with just a filename need to be in the same folder as your script

## Program Dog\_mir\_find.py

mir\_File=open('mature.fa',mode='rU') ## Open an existing file to **r**ead

mir\_seqList=[] ## Start with an empty list, to fill with mir sequences

mir\_nameList=[] ## Another empty list to fill with mir names

for L1 in mir\_File: ## Go through every line in the file

L1stripped=L1.strip()## Start by stripping white space from the line (returns, spaces at ends)

if L1stripped[0]=='>': ## Inspection of the file reveals that each name is prefaced by a '>' (called FastA format)

mir\_name=L1stripped[1:] ## For the moment, take everything but the first '>' character as the name

else:

mir\_seq=L1stripped.replace('U','T') ## If it's not a name, it's a sequence. Replace "U" with "T" since we're matching to DNA

if 'Canis' in mir\_name: ## This is a dog experiment, take only sequences where the name includes 'Canis'

mir\_seqList.append(mir\_seq) ## Extend the list of sequences, adding the current sequence

mir\_nameList.append(mir\_name.split(' ')[0]) ## Extend name list, .split(' ')[0] generates the first word of name

mir\_File.close() ## Close the file

read\_File=open('myDogsRNA.txt',mode='rU') ## Open the read file

mir\_num=len(mir\_nameList) ## The length of a list is the number of elements in that list. This tells us how many times to loop below

mir\_hitList=[0]\*mir\_num ## This generates a list with mir\_num instances of 0 (e.g., [0]\*9 would be [0,0,0,0,0,0,0,0,0]

for L2 in read\_File: ## A loop that looks for individual reads in the miRNA list and increments the appropriate item in mir\_hitList

poslinker=L2.rfind('TCGT') ## find the first occurence of the end of the linker (start from the right side)

if poslinker==-1: ## if no occurence of the beginning of the linker was found

continue ## jump to the next "Run-Through" of the current loop (effectively skipping this small RNA read)

L2delinked=L2[:poslinker] ## just take the read before (but not including) the first base of the presumed linker

if L2delinked in mir\_seqList:

mir\_hit=mir\_seqList.index(L2delinked)

mir\_hitList[mir\_hit]=mir\_hitList[mir\_hit]+1

read\_File.close()

out\_File=open('myDogmiRNAhits.csv',mode='w') ## Output file (opened in write mode)

for n1 in range(mir\_num): ## A loop that prints the relevant values

if mir\_hitList[n1]>0:

out\_File.write(mir\_nameList[n1]+','+ mir\_seqList[n1]+','+str(mir\_hitList[n1])+'\r') ## separate with commas

out\_File.close()

**2b.** Dogs can get systemic nematode infections ('arf!'). Generate a list of sequences in " myDogsRNA.txt " that are absent in the list of Dog miRNAs but present in the list of C. elegans miRNAs.

Your code:

Paste your code here

Your answer (list of sequences from myDogsRNA.txt that match elegans but not dog miRNA :

Paste your output here

**2c.** If your list is not empty, maybe there are some miRNAs shared between vertebrates and worms but somehow not annotated yet in the dog genome. Make a list of miRNAs that are present in " myDogsRNA.txt " that are in the three worm genomes represented in mature.fa (elegans, remanei, and briggsae) but absent in both humans and dogs.

Your code:

Paste your code here

Your answer (sequences in myDogsRNA.txt matching elegans/briggsiae/remanei but not dog/human miRNA)

Paste your output here

**2d. (Optional)** Ouch, if your list is still not empty, does this mean that Fido has a nematode infection? Other possible explanations?

Your answer (thoughts and ideas and inferences on what could be going on with Fido)

Paste your thoughts/ideas/inferences here

**Problem 3. Some fun with E. coli:**

**3a1.** The sequence ColiDH1.fa contains the sequence of E. coli strain DH1. Determine the content of A, C, G, and T bases in this genome

## Base\_Composition.py

F=open('ColiDH1.fa',mode='rU') ## standard way to open a file

Seq1='' ## start with an empty string

for L in F: ## loop through lines in the file

if L=='' or L[0]=='>': ## for any line that is blank or starts with '>' (i.e., sequence description)

continue ## ignore this line

Seq1+=L.strip() ## any other line, add to the Sequence (note the use of "+=". **foo+=bar** is a just a quick-and-easy way to abbreviate **foo=foo+bar**)

F.close() ## close the file

bases1=['A','C','T','G'] ## make a list of bases that we'll loop through

for b1 in bases1: ## loop through the list of four bases

print b1+'\t'+str(Seq1.count(b1)) ## for each base, Output a line with the base name and number of hits in sequence.

Answers (If you haven't done this yet, run the above program on your system and report what results arise):

A,\_\_\_\_\_\_\_\_ C, \_\_\_\_\_\_\_\_ G, \_\_\_\_\_\_\_\_ T, \_\_\_\_\_\_\_\_

**3a2.** Now count the occurence of all dinucleotide pairs (e.g., "AA","AC","AG","AT"...) Your output should be in the form of 16 consecutive lines, each with a sequence, a tab, and # of occurences in the genome.

Should we use the "string.count(substring)" method for counting dinucleotides? Do a quick "reality check" with this: count the number of A's in "AAAA". Now count the number of AA's in "AAAA". Do we want this answer? <**No**, so use a different approach of going through the sequence dinucleotide by dinucleotide>

## DiNucleotide\_Composition.py

F1=open('ColiDH1.fa',mode='rU') ## standard way to open a file

Seq1='' ## start with an empty string

for L in F1: ## loop through lines in the file

if L=='' or L[0]=='>': ## for any line that is blank or starts with '>' (i.e., sequence description)

continue ## ignore this line

Seq1+=L.strip() ## any other line, add to the Sequence (note "A+=B" is a just abbreviation for A=A+B)

F1.close() ## close the file

bases1=['A','C','T','G'] ## make a list of bases that we'll loop through

dinucleotide\_sequences1=[] ## an array that will contain all sixteen dinucleotide sequences

dinucleotide\_instances1=[0]\*16 ## an array that will contain numbers of occurences for each dinucleotide

for b1 in bases1: ## loop through the list of four bases

for b2 in bases1: ## loop through the list of four bases

d1=b1+b2 ## d1 is the current dinucleotide

dinucleotide\_sequences1.append(d1) ## append d1 to the dinucleotide list

lenSeq1=len(Seq1) ## how many times through the loop for E. coli DH1

Seq1+=Seq1[:5] ## Bacterial chromosome is a circle, so put the first 5 bases of Seq1 after Seq1.

for i1 in range(lenSeq1): ## let's go through this loop for each base in the E. coli DH1 genome

SeqWord1=Seq1[i1:i1+2] ## Slice out a sequence starting at base i1, with length 2

Seq\_index1=dinucleotide\_sequences1.index(SeqWord1) ## What is the index of the current dinucleotide in the dinucleotide array

dinucleotide\_instances1[Seq\_index1]+=1 ## Increment the count of that particular dinucleotide

for i2 in range(16): ## Output the index, sequence, and number of hits for each

print str(i2)+'\t'+dinucleotide\_sequences1[i2]+'\t'+str(dinucleotide\_instances1[i2])

##note the *str()* is needed to ensure strings are added to each other (not strings plus numbers)

Your answer (the output of the preceeding code: counts of each of the 16 dinucleotides in the Coli genome)

Paste your output here

**3b.** Okay, now that we're warmed up, lets do all possible 4-mers. Do the output as above. Also import the data into excel or another spreadsheet and sort the tetranucleotides and occurences by the occurence in the E. coli genome. What is the most under-represented tetranucleotide? Second-most-under-represented?

Your code:

Paste your code here

Your answer (list of tetranucleotide sequences and counts in Coli genome [sort by counts using a spreadsheet]

Paste your output here

**3c.** Come up with a hypothesis, or two, or three to potentially explain the dramatic underrepresentation of this tetranucleotide.

Your thoughts/ideas (and at least one wild guess)

Paste your thoughts/ideas/inferences here

**3d.** (Optional) Look in the two small RNA sequence files (the dog-derived small RNA sequences in **myDogsRNA.txt** and the worm-derived small RNA sequences in **fastQfileFiltered.txt**) for segments that reside in the E. coli genome and are longer than 21bp. What could these sequences mean?

Your code:

Paste your code here

Your answer (list of >21bp sequences from Coli in the worm and dog sequence reads

Paste your output here

Your thoughts/ideas

Paste your thoughts/ideas/inferences here