Code:

The following is the code used to put together the hydropathy plots

```
#usr/bin/python33
import scipy
import numpy
import matplotlib
import re
import Bio
#import os
# import tkinter will implement this one later
import fasta_reader
from matplotlib import pylab
from numpy import *
from pylab import *

# This program will ask for file names.
def theopener():
        global cinput
        cinput = input("Enter a file name: ")
        cifile = open(cinput)
        global cirecord
        cirecord = fasta_reader.read_fasta_record(cifile)
        return (cirecord)
        return (cinput)

#This program will count the frequency of each amino acid (or nucleotide) in the sequence

def seqcount():
        from collections import Counter
        seqcountinput = input("Would you like to examine a file (f) or sequence (s)? ")
        if seqcountinput == "f":
                theopener()
                mer = len(cirecord.sequence)
                parray = []
                for i in range(0, mer):
                        parray.append(cirecord.sequence[i])
                protarray = array(parray)
                cat = Counter(protarray)
                print(cat)
        if seqcountinput == "s":
                seqcountseq = input("Input your sequence: ")
                cat = Counter(seqcountseq)
                print(cat)

# This program will search a sequence for a particular string
def seqregex():
        theopener()
        rxput = input("What is the sequence you would like to match against this file? ")
        regmatch = re.search((rxput), (cirecord.sequence))
        if regmatch:
                print('\n Found', regmatch.group(),
cirecord.sequence.count(regmatch.group()), 'times. \n')
        else:
                print('\n Found no matches.')

# This program will create hydropathy plots for a given protein
def hydropathy():
        plotputs = input("Would you like to plot 1, 2, or 3 proteins? ")
        if plotputs == "1":
                # ask how long sequences and overlapping windows should be, set variables
                theopener()
                record_id = cirecord.title.split()[0]
                reslength = input("\nHow many residues do you want your plots to have? ")
                reslengths = int(reslength)
                overlapq = input("\nHow long of an overlap between plots do you want? ")
                overlapz = int(overlapq)
                leftover = (reslengths-overlapz)
                smooths = input("\nHow many residues do you want to average for smoothing?
                \n(Enter 1 for no smoothing) ")
                window = int(smooths)
```

```
        # Get lengths of sequences, create new variables
        ciresidues = len(cirecord.sequence)
        civalues = []
        # Match regular expression to filename to be able to save files later
        cisave = cinput.split(".", 1)
        cisaves = cisave[0]
        # Enter hydrophobicity values as dictionary
        kd = { 'A': 1.8,'R':-4.5,'N':-3.5,'D':-3.5,'C': 2.5,
               'Q':-3.5,'E':-3.5,'G':-0.4,'H':-3.2,'I': 4.5,
               'L': 3.8,'K':-3.9,'M': 1.9,'F': 2.8,'P':-1.6,
               'S':-0.8,'T':-0.7,'W':-0.9,'Y':-1.3,'V': 4.2 }

        # Append values from hydrophobicity dictionary to fasta sequences in new
        variable
        for residue in cirecord.sequence:
               civalues.append(kd[residue])

        # Create window for smoothing that will work with value of 1 in case
        smoothing is not desired
        half_window = int(trunc((window-1)/2))
        sm_data = []
        # Add function to smooth if required
        for p in range(half_window, ciresidues-half_window):
               average_value = 0.0
               for q in range(-half_window, half_window+1):
                      average_value += civalues[p+q]
               sm_data.append(average_value / window)
        # Create new variables for plotting
        smciseq = []
        # Create list of lists to get overlapping AA windows, turn to dictionary
        so pyplot can read it
        smciseq = [sm_data[d:d+reslengths] for d in range(1, len(sm_data),
        (leftover))]
        smciseqarray = np.array(smciseq)
        print("This will create", len(smciseqarray), "files of",
        len(smciseqarray[1]), "residues in length.")
        #this creates the actual plots
        for i in range(0, len(smciseqarray)):
               for j in (0, len(smciseqarray[i])):
                      axis(xmin=1, xmax=reslengths, ymin=-7, ymax=7)
                      x_data = range(1, 1+len(smciseqarray[i]))
                      plot(x_data, smciseqarray[i], linewidth=1.0)
                      xlabel("Residue")
                      ylabel("Hydropathy")
                      title("Hydropathy Plot of "+cisaves+" \n starting  at amino
                      acid residue "+str((i*leftover)+1))
                      savefig("%s_%04d.png" % (cisaves, (i)))
                      close()
if plotputs == "2":
        theopener()
        #gets name of second file, splits title for file usage
        secinput = input("What is the name of your second file? ")
        secfile = open(secinput)
        secrecord = fasta_reader.read_fasta_record(secfile)
        record_id = cirecord.title.split()[0]
        secord_id = secrecord.title.split()[0]
        ciresidues = len(cirecord.sequence)
        secresidues = len(secrecord.sequence)

        civalues = []
        secvalues = []
        # Match regular expression to filename to be able to save files later
        cisave = cinput.split(".", 1)
        cisaves = cisave[0]
        secsave = secinput.split(".",1)
        secsaves = secsave[0]
        ciresidues = len(cirecord.sequence)
        secresidues = len(secrecord.sequence)
        #Window function for hydrophobicity plots
        reslength = input("\nHow many residues do you want your plots to have? ")
```

```
reslengths = int(reslength)
overlapq = input("\nHow long of an overlap between plots do you want? ")
overlapz = int(overlapq)
leftover = (reslengths-overlapz)
smooths = input("\nHow many residues do you want to average for smoothing?
\n(Enter 1 for no smoothing) ")
window = int(smooths)

# Enter hydrophobicity values as dictionary
kd = { 'A': 1.8,'R':-4.5,'N':-3.5,'D':-3.5,'C': 2.5,
       'Q':-3.5,'E':-3.5,'G':-0.4,'H':-3.2,'I': 4.5,
       'L': 3.8,'K':-3.9,'M': 1.9,'F': 2.8,'P':-1.6,
       'S':-0.8,'T':-0.7,'W':-0.9,'Y':-1.3,'V': 4.2 }

for residue in cirecord.sequence:
       civalues.append(kd[residue])

for residue in secrecord.sequence:
       secvalues.append(kd[residue])

half_window = int(trunc((window-1)/2))
sm_data = []
se_data = []
# Add smoothing function
for p in range(half_window, ciresidues-half_window):
       average_value = 0.0
       average_seval = 0.0
       for q in range(-half_window, half_window+1):
              average_value += civalues[p+q]
              average_seval += secvalues[p+q]
       sm_data.append(average_value / window)
       se_data.append(average_seval / window)

smciseq = []
smsecseq = []
# Create list of lists to get overlapping AA windows, turn to dictionary
so pyplot can read it
smciseq = [sm_data[d:d+reslengths] for d in range(1, len(sm_data),
(leftover))]
smciseqarray = np.array(smciseq)
smsecseq = [se_data[d:d+reslengths] for d in range(1, len(se_data),
(leftover))]
smsecseqarray = np.array(smsecseq)
if len(smciseqarray) > len(smsecseqarray):
       shortseq = len(smsecseqarray)
       shortseqq = int(shortseq)
else:
       shortseq = len(smciseqarray)
       shortseqq = int(shortseq)

#create heat map plots
ciseq, secseq = np.meshgrid(civalues, secvalues)
ciplot = abs(ciseq - secseq) <= 0.3
ciplot2 = abs(ciseq - secseq)
imshow(ciplot, cmap=cm.autumn)
colorbar()
#plot([50, ciresidues-50], [50, secresidues-50], color="white",
linewidth=0.5)
axis(xmin=0, xmax=ciresidues, ymin=0, ymax=secresidues)
xlabel(cisaves+" Residue")
ylabel(secsaves+" Residue")
title("Hydropathy Comparison of "+cisaves+" and "+secsaves)
savefig('%s_%s_equal.png' % (cisaves, secsaves))
clf()
imshow(ciplot2)
colorbar()
axis(xmin=0, xmax=ciresidues, ymin=0, ymax=secresidues)
xlabel(cisaves+" Residue")
ylabel(secsaves+ "Residue")
title("Hydropathy Comparison of"+cisaves+" and "+secsaves)
savefig('%s_%s_difference.png' % (cisaves, secsaves))
```

```
        clf()
        #creates line graph plots with two lines
        for i in range(0, shortseqq):
                for j in (0, len(smciseqarray[i])):
                        axis(xmin=1, xmax=reslengths, ymin=-7, ymax=7)
                        x_data = range(1, 1+len(smciseqarray[i]))
                        plot(x_data, smciseqarray[i], x_data, smsecseqarray[i],
                        linewidth=1.0)
                        xlabel("Residue")
                        ylabel("Hydropathy")
                        title("Hydropathy Plot of "+cisaves+ " and" +secsaves+ "\n
                        starting  at amino acid residue "+str((i*leftover)+1))
                        savefig("%s_%s_%04d.png" % (cisaves, secsaves, (i+1)))
                        clf()
if plotputs == "3":
        theopener()
        #gets name of second file, splits title for file usage
        secinput = input("What is the name of your second file? ")
        secfile = open(secinput)
        secrecord = fasta_reader.read_fasta_record(secfile)
        thirdput = input("What is the name of your third file? ")
        thirfile = open(thirdput)
        thirecord = fasta_reader.read_fasta_record(thirfile)
        record_id = cirecord.title.split()[0]
        secord_id = secrecord.title.split()[0]
        thircord_id = thirecord.title.split()[0]
        ciresidues = len(cirecord.sequence)
        secresidues = len(secrecord.sequence)
        thiresidues = len(thirecord.sequence)

        civalues = []
        secvalues = []
        thivalues = []
        # Match regular expression to filename to be able to save files later
        cisave = cinput.split(".", 1)
        cisaves = cisave[0]
        secsave = secinput.split(".",1)
        secsaves = secsave[0]
        thisave = thirdput.split(".", 1)
        thisaves = thisave[0]
        ciresidues = len(cirecord.sequence)
        secresidues = len(secrecord.sequence)
        thiresidues = len(thirecord.sequence)
        #Window function for hydrophobicity plots
        reslength = input("\nHow many residues do you want your plots to have? ")
        reslengths = int(reslength)
        overlapq = input("\nHow long of an overlap between plots do you want? ")
        overlapz = int(overlapq)
        leftover = (reslengths-overlapz)
        smooths = input("\nHow many residues do you want to average for smoothing?
        \n(Enter 1 for no smoothing) ")
        window = int(smooths)

        # Enter hydrophobicity values as dictionary
        kd = { 'A': 1.8,'R':-4.5,'N':-3.5,'D':-3.5,'C': 2.5,
                'Q':-3.5,'E':-3.5,'G':-0.4,'H':-3.2,'I': 4.5,
                'L': 3.8,'K':-3.9,'M': 1.9,'F': 2.8,'P':-1.6,
                'S':-0.8,'T':-0.7,'W':-0.9,'Y':-1.3,'V': 4.2, "-": 0.0, }

        for residue in cirecord.sequence:
                civalues.append(kd[residue])

        for residue in secrecord.sequence:
                secvalues.append(kd[residue])

        for residue in thirecord.sequence:
                thivalues.append(kd[residue])

        half_window = int(trunc((window-1)/2))
        sm_data = []
        se_data = []
```

```
                th_data = []
                # Add smoothing function
                for p in range(half_window, ciresidues-half_window):
                        average_value = 0.0
                        average_seval = 0.0
                        average_thival = 0.0
                        for q in range(-half_window, half_window+1):
                                average_value += civalues[p+q]
                                average_seval += secvalues[p+q]
                                average_thival += thivalues[p+q]
                        sm_data.append(average_value / window)
                        se_data.append(average_seval / window)
                        th_data.append(average_thival / window)

                smciseq = []
                smsecseq = []
                smthiseq = []
                # Create list of lists to get overlapping AA windows, turn to dictionary
                so pyplot can read it
                smciseq = [sm_data[d:d+reslengths] for d in range(1, len(sm_data),
                (leftover))]
                smciseqarray = np.array(smciseq)
                smsecseq = [se_data[d:d+reslengths] for d in range(1, len(se_data),
                (leftover))]
                smsecseqarray = np.array(smsecseq)
                smthiseq = [th_data[d:d+reslengths] for d in range(1, len(th_data),
                (leftover))]
                smthiseqarray = np.array(smthiseq)
                if len(smciseqarray) > len(smsecseqarray):
                        shortseq = len(smsecseqarray)
                        shortseqq = int(shortseq)
                else:
                        shortseq = len(smciseqarray)
                        shortseqq = int(shortseq)

                #creates line graph plots with three lines
                for i in range(0, shortseqq):
                        for j in (0, len(smciseqarray[i])):
                                axis(xmin=1, xmax=reslengths, ymin=-7, ymax=7)
                                x_data = range(1, 1+len(smciseqarray[i]))
                                plot(x_data, smciseqarray[i], x_data, smsecseqarray[i],
                                x_data, smthiseqarray[i], linewidth=1.0)
                                xlabel("Residue")
                                ylabel("Hydropathy")
                                title("Hydropathy Plot of "+cisaves+ " and" +secsaves+ "\n
                                starting  at amino acid residue "+str((i*leftover)+1))
                                savefig("%s_%s_%s_%04d.png" % (cisaves, secsaves,
                                thisaves,(i+1)))
                                clf()
                return

def getncbi():
        #imports necessary modules
        from Bio import Entrez
        from Bio import SeqIO
        email = input("What is the email address you would like to use to query Entrez? ")
        Entrez.email = email
        handle = input("Would you like to search for a nucleotide (n) or protein (p)? ")
        #for nucleotide sequence retrieval
        if handle == "n":
                entsone = 'nuccore'
                ginput = input("What is the gi number of your nucleotide sequence? ")
                sqandle = Entrez.efetch(db=entsone, id=ginput, rettype="fasta",
                retmode="text")
                handleread = SeqIO.read(sqandle, "fasta")
                handlestring = str(handleread.seq)
                handlesplit = handleread.description.split(sep="|")
                handid = handlesplit[1]
                handids = str(handid)
                clascat = ">"+handleread.description+"\n"+handlestring
                output = open(handids+".txt", "w")
```

```
                output.write(clascat)
                output.close()
                print("The fasta sequence is in %s.txt" % handids)
                gotostart()
        #for protein sequence retrieval
        if handle == "p":
                entsone = 'protein'
                ginput = input("What is the gi number of your protein sequence? ")
                sqandle = Entrez.efetch(db=entsone, id=ginput, rettype="fasta",
                retmode="text")
                handleread = SeqIO.read(sqandle, "fasta")
                handlestring = str(handleread.seq)
                handlesplit = handleread.description.split(sep="|")
                handid = handlesplit[1]
                handids = str(handid)
                clascat = ">"+handleread.description+"\n"+handlestring
                output = open(handids+".txt", "w")
                output.write(clascat)
                output.close()
                print("The fasta sequence is in %s.txt" % handids)
                gotostart()

def gotostart():
        originput =  input("\n 1. Monomer Frequency Counter \n 2. Search for a Subsequence
\n 3. Hydropathy Plot \n 4. Get Fasta Sequence from Database \n\n Which program would you
like to run? ")
        if originput == "1":
                seqcount()

        if originput == "2":
                seqregex()

        if originput == "3":
                hydropathy()

        if originput == "4":
                getncbi()

gotostart()
```

The following is the code used to determine correlative values

```
#usr/bin/python33
#import modules for program
import scipy
import numpy as np
import math
import fasta_reader

#hydropathy dictionary
kd = { 'A': 1.8,'R':-4.5,'N':-3.5,'D':-3.5,'C': 2.5,
    'Q':-3.5,'E':-3.5,'G':-0.4,'H':-3.2,'I': 4.5,
    'L': 3.8,'K':-3.9,'M': 1.9,'F': 2.8,'P':-1.6,
    'S':-0.8,'T':-0.7,'W':-0.9,'Y':-1.3,'V': 4.2 }

like = { 'A': 2.78,'R': -3.87,'N': -0.81,'D':-3.5,'C': -0.81,
    'Q': -0.81,'E':-3.5,'G': 2.78,'H': -3.87,'I': 2.78,
    'L': 2.78,'K': -3.87,'M': -0.81,'F': 0.2,'P': -0.81,
    'S': -0.81,'T': -0.81,'W': 0.2,'Y': 0.2,'V': 2.78 }

#get files
cinput = input("What is the name of your first file? ")
cifile = open(cinput)
sinput = input("What is the name of your second file? ")
secfile = open(sinput)
cirecord = fasta_reader.read_fasta_record(cifile)
secrecord = fasta_reader.read_fasta_record(secfile)
civalues = []
secvalues = []
for residue in cirecord.sequence:
    civalues.append(kd[residue])
```

```
for residue in secrecord.sequence:
    secvalues.append(kd[residue])
cipositives = []
for residue in cirecord.sequence:
    cipositives.append(like[residue])
secpositives = []
for residue in secrecord.sequence:
    secpositives.append(like[residue])

#define the average
def average(x):
    assert len(x) > 0
    return float(sum(x)) / float(len(x))

#define pearson coefficient
def pearson_r(x, y):
    assert len(x) == len(y)
    n = len(x)
    assert n > 0
    avg_x = average(x)
    avg_y = average(y)
    diffprod = 0
    xdiff2 = 0
    ydiff2 = 0
    for dx in range(n):
        xdiff = x[dx] - avg_x
        ydiff = y[dx] - avg_y
        diffprod += (xdiff * ydiff)
        xdiff2 += (xdiff * xdiff)
        ydiff2 += (ydiff * ydiff)
    sqrdiffprod = (xdiff2 * ydiff2)**(1/2)
    pears_r = diffprod / sqrdiffprod
    print(diffprod)
    print(xdiff2 * ydiff2)
    print(sqrdiffprod)
    print("The Pearson R value is ", pears_r)
    pears_r2 = (pears_r ** 2)
    print ("The R-squared value is ", pears_r2)

print("For identity of your data sets: ")
pearson_r(civalues, secvalues)
print("For positives in your data sets: ")
pearson_r(cipositives, secpositives)
```

Previous Work:

Before undertaking this research , I had previously been working on a different experiment. The purpose of this research was to investigate the interactions of the *DllA/DllB* bigene cluster in *Ciona*. These two genes, orthologs of the Distal-less gene in fruit flies, are expressed at different times in varying places during *Ciona* embryogenesis. My research was intended to solely focus on expression in the ectoderm. Much like the current experiment, *Ciona* was being used to model a simplified pathway, as the *DllA/DllB* genes are arranged the same way as the Dlx cluster is in humans (Irvine, 2007), the difference being humans have 3 such clusters, and *Ciona* 1.

The first step in this process was to put together a 'double reporter' construct, a gene that could fluoresce if either DllA or DllB was being expressed. This process involved splicing two plasmids that could each fluoresce for one of these proteins (L1000 and L1040 in the figures below) into one that could fluoresce for both (L1060) and cloning it in *Escherichia coli*. This proved to be far more difficult than anticipated.

All attempts to successfully create the double-reporter gene ended in failure, generally somewhere between the ligation of the L1000 and L1040 plasmids and the cloning of the bacteria containing L1060. No colonies containing the plasmid developed.

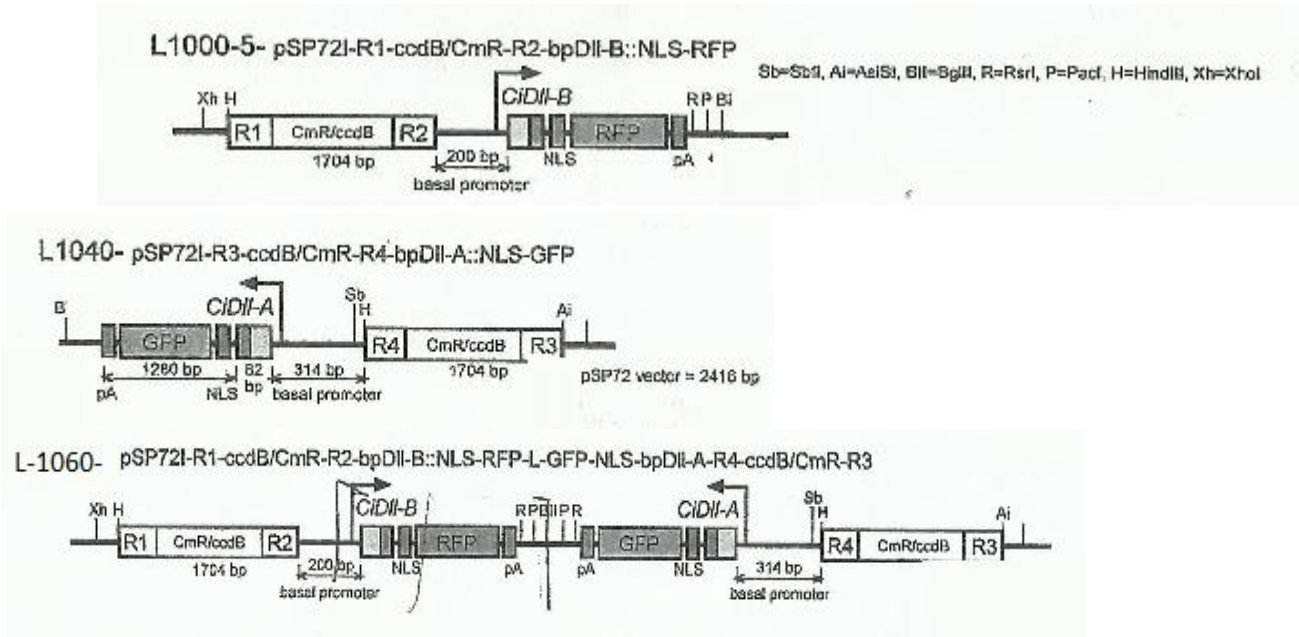

Fig S-PW – The theoretical Double Reporter construct and its components, with restriction sites (Xh, H, Sb, Ai), recombination sites (R1 – R4), the Ciona distal-less A and B regions, and the green and red fluorescent proteins. 7a – L1000, the DllB component; 7b – L1040, the DllA component; 7c – Double Reporter gene construct.