March 8, 2012

Downloaded "DJ-M\_BBTMALL.txt" from an e-mail from Daniel.

Copied "\* without pdb.clu" and "\*.fasta" from "test moments folder"; info on how they were originally optained is available in that folder's log.

Created "\* with gonnet.aln" by using ClustalX to append "\*.fasta" to "\* without pdb.clu" and aligning, in ClustalX with the matrix set to the "Gonnet series" option.

I created "\* with bbtm.aln" similarly, but choosing the "User Defined option", with "DJ-M\_BBTMALL.txt" as the matrix.

Looking through the files, I saw that the “Scry \*.aln” alignments don't have the “scry\_chaina” sequence from “scry.fasta”; I must have made a mistake. “OMPLA \*.aln” alignments have “chaina\_ ompla”

March 9, 2012

Opened OMPLA in PyMOL. Observed that the segment FTLYPYD begins at position 0. Colored residue 30 blue to verify that the phenylalinine in FTLY is 0.

Used the check\_range function of the following script:

from \_\_future\_\_ import division

from \_\_future\_\_ import print\_function

import sys

sys.path.append("/home/davis/work stuff/beta-barrel-oligomerization/modules")

import numpy as np

import zenergy

import csv

import selections

from functools import partial

import Bio.AlignIO

from sundries import CIDict

from sundries import file\_dict

from Bio.PDB import PDBParser

import warnings

import os

import matrices

import math

class MissingPDBSequence(Exception):

pass

class MultiplePDBSequences(Exception):

pass

class AlignmentOracle(object):

'''

Initialized with a multiple sequence alignment and a name to find a

particular sequence of interest. Can give amino acids from other

sequences in the alignment that correspond to amino acids in the

sequence

of interest, ignoring amino acids that are lined up with gaps in the

sequence of interest.

'''.replace(' ', '')

def \_\_init\_\_(self, alignment, pdb\_name = 'pdb'):

self.data = alignment

self.pdb\_index = None

for index, record in enumerate(self.data):

if pdb\_name in record.id:

if self.pdb\_index is None:

self.pdb\_index = index

else:

raise MultiplePDBSequences('More than one sequence had '+\

pdb\_name + ' in its title')

if self.pdb\_index == None:

raise MissingPDBSequence('None of the sequences had '+\

pdb\_name + ' in their title')

self.\_pdb\_sequence = self.data[self.pdb\_index].seq

def pdb\_sequence(self, selection = None):

return self.sequence(self.pdb\_index, selection)

def sequence(self, index, selection = None):

pos = 0

for j, letter in enumerate(self.data[index].seq):

if self.\_pdb\_sequence[j] != '-':

if selection is None or pos in selection:

yield letter.upper()

pos += 1

def get\_alignment(self):

return self.data

def get\_pdb\_index(self):

return self.pdb\_index

def get\_pdb\_seq\_record(self):

return self.data[self.pdb\_index]

# Create dictionaries that map names of proteins to other dictionaries

bbtm\_alignments = dict()

gonnet\_alignments = dict()

for protein in ('OMPLA',):

# Create dictionaries that map names of sequences to sequences containing

# only the positions that are aligned to non-gap positions in the pdb

# sequence

abridged\_dicts = list()

for filename in (protein + ' bbtm.aln',

protein + ' gonnet.aln'):

alignment = Bio.AlignIO.read(filename, 'clustal')

oracle = AlignmentOracle(alignment, pdb\_name = 'chaina')

abridged = dict()

for i in range(len(alignment)):

abridged.update({oracle.data[i].id: (''.join(oracle.sequence(i)))})

abridged\_dicts.append(abridged)

bbtm\_alignments.update({protein: abridged\_dicts[0]})

gonnet\_alignments.update({protein: abridged\_dicts[1]})

def check\_range(start, end, protein):

# Observe what positions of each protein in the gonnet alignment and the

# bbtm alignment correspond to a given range in the pdb sequence.

# Positions ocrresponding to gaps in the pdb sequence not shown.

print('name then bbtm segment then gonnet segment')

for name in bbtm\_alignments[protein].keys():

print(name)

print(bbtm\_alignments[protein][name][start: end])

print(gonnet\_alignments[protein][name][start:end])

print()

check\_range(0,10) includes in its output:

chaina\_ompla

FTLYPYDTNY

FTLYPYDTNY

chiana\_ompla is the name of the pdb sequence in “OMPLA.fasta”

So position 0 in the sequences correspond with position 30 in PyMOL, for OMPLA.

Copied “identity.csv”, an identity matrix, from “test moments/identity.csv”

Changed the definition of check\_range to

with open('identity.csv', 'rb') as f:

id\_matrix = matrices.retrieve\_matrix(csv.reader(f))

def check\_range(start, end, protein):

# Observe what positions of each protein in the gonnet alignment and the

# bbtm alignment correspond to a given range in the pdb sequence.

# Positions ocrresponding to gaps in the pdb sequence not shown.

print('name then seq distance then bbtm segment then gonnet segment')

pdb\_seq\_name = 'chaina\_' + protein.lower()

for name in bbtm\_alignments[protein].keys():

print(name)

print(matrices.compare(bbtm\_alignments[protein][name],

bbtm\_alignments[protein][pdb\_seq\_name],

id\_matrix))

print(matrices.compare(gonnet\_alignments[protein][name],

gonnet\_alignments[protein][pdb\_seq\_name],

id\_matrix))

print(bbtm\_alignments[protein][name][start: end])

print(gonnet\_alignments[protein][name][start:end])

print()

One strand in PyMOL stretches from PyMOL coordinates 195-202, in sequence coordinates 165-172. I ran check\_range(165,172,'OMPLA'), saved in file “165 172.txt”. Selections:

gi|29653831|ref|NP

0.279166666667

0.358333333333

GRVVFA-

GRVVFAV

gi|85060316|ref|YP

0.5125

0.745833333333

YRLKIG-

YRLKIGY

Another strand is PyMOL 206-214, sequence 176-184. From check\_range(176,184,'OMPLA'), saved in “176 184.txt”:

gi|53831|ref|NP

0.279166666667

0.358333333333

H-RQVLSL

QVLSLMLR

gi|85060316|ref|YP

0.5125

0.745833333333

G-ESIFSA

SIFSAEGR

Those were both non-interface strands. An interface strand is PyMOL 86-98, sequence 56-68. From check\_range(56, 68, 'OMPLA'), saved in “56 68.txt”:

gi|29653831|ref|NP

0.279166666667

0.358333333333

S-LDISYTQLSY

LSLDISYTQLSY

gi|85060316|ref|YP

0.5125

0.745833333333

L-LGASYTQRSW

SLLGASYTQRSW