# August 22, 2012

The following is code to calculate zdiff's. It's been tested; the details are in "june 2012 lab meeting log".

**from** **sundries** **import** CIDict

**from** **Bio.PDB** **import** PDBParser

**import** **warnings**

**from** **Bio** **import** AlignIO

**def** z(residue):

*'''Returns the z coordinate of a residue object's Calpha.'''*

**return** residue.child\_dict['CA'].get\_coord()[2]

**class** **Gap**(object):

*'''Represents a gap in a Position object.'''*

**pass**

**class** **Position**(object):

**def** \_\_init\_\_(self, pairs):

self.residues = CIDict(pairs)

**def** zdiff(self, template\_id, unknown\_id):

template\_res = self.residues[template\_id]

unknown\_res = self.residues[unknown\_id]

**if** type(template\_res) **is** Gap **or** type(unknown\_res) **is** Gap:

**return** None

**else**:

**return** z(template\_res) - z(unknown\_res)

**def** resi(self, stru\_id):

**return** self.residues[stru\_id].get\_id()[1]

**class** **NotFoundError**(Exception):

**pass**

**class** **Zdiff**(object):

**def** \_\_init\_\_(self, \*stru\_seq\_pairlist):

pos\_inputs = list()

**for** structure, sequence **in** stru\_seq\_pairlist:

residues = structure.get\_residues()

pairs\_for\_pos = list()

**for** letter **in** sequence:

**if** letter == '-':

seq\_unit = Gap()

**else**:

seq\_unit = residues.next()

pairs\_for\_pos.append((structure.get\_id(), seq\_unit))

pos\_inputs.append(pairs\_for\_pos)

self.positions = [Position(id\_res\_pairlist) \

**for** id\_res\_pairlist **in** zip(\*pos\_inputs)]

**def** get(self, template\_id, unknown\_id, resi, start=0):

**for** pos **in** self.positions[start:]:

**if** pos.resi(unknown\_id) == resi:

**return** pos.zdiff(template\_id, unknown\_id)

**raise** NotFoundError('resi {} of {} not found' \

.format(resi, unknown\_id))

**def** report(self, template\_id, unknown\_id):

output = (pos.zdiff(template\_id, unknown\_id) \

**for** pos **in** self.positions)

**return** filter(**lambda** x: x **is** **not** None, output)

**with** warnings.catch\_warnings():

warnings.simplefilter('ignore')

scry\_stru = PDBParser().get\_structure('1A0S', 'aligned\_1A0S.pdb')

maltoporin\_stru = PDBParser().get\_structure('1AF6', 'aligned\_1AF6.pdb')

alignment = AlignIO.read('Swiss-PDB structural alignment.aln', 'clustal')

**for** seq\_record **in** alignment:

**if** seq\_record.id == 'aligned\_1A0S':

scry\_seq = seq\_record

**if** seq\_record.id == 'aligned\_1AF6':

maltoporin\_seq = seq\_record

output = Zdiff((scry\_stru, scry\_seq), (maltoporin\_stru, maltoporin\_seq))

**with** open('NEW 1a0s as template.txt', 'w') **as** f:

f.writelines(str(zdiff) + '**\n**' \

**for** zdiff **in** output.report('1A0S', '1AF6'))

**with** open('NEW 1af6 as template.txt', 'w') **as** f:

f.writelines(str(zdiff) + '**\n**' \

**for** zdiff **in** output.report('1af6', '1a0s'))

My current goal is to turn this code into a module that can be accessed with easy-to-remember commands, and check that this module can reproduce the old results.

Copying some of the data from "june 2012 lab meeting" so I can try to replcate it. From "june 2012 lab meeting/zdiff calculator validation", I copied two structures, an alignment, and a list of zdiffs. The list of zdiffs does not have resi's, so I am pretending that I know that it is in order by resi. These files are:  
aligned\_1A0S.pdb  
aligned\_1AF6.pdb  
OLD 1a0s as template.txt  
Swiss-PDB structural alignment.aln

They all have their original names, except for "OLD 1a0s as template.txt", which used to be "NEW 1a0s as template.txt".

Remade the zdiff's with new code. The zdiff's at the top and bottom came out correct, and there was the right number of them, so I figure it probably works.

The code I used was this:

**from** sundries **import** CIDict

**from** Bio.PDB **import** PDBParser

**import** warnings

**from** Bio **import** AlignIO

# The guts that it runs on

**def** z(residue):

'''Returns the z coordinate of a residue object's Calpha.'''

**return** residue.child\_dict['CA'].get\_coord()[2]

**class** Gap(object):

'''Represents a gap in a Position object.'''

**pass**

**class** Position(object):

**def** \_\_init\_\_(self, pairs):

self.residues = CIDict(pairs)

**def** zdiff(self, template\_id, unknown\_id):

template\_res = self.residues[template\_id]

unknown\_res = self.residues[unknown\_id]

**if** type(template\_res) **is** Gap **or** type(unknown\_res) **is** Gap:

**return** None

**else**:

**return** z(template\_res) - z(unknown\_res)

**def** resi(self, stru\_id):

'''Return resi of position in specified structure'''

# Get the residue of the specified structure that is in this

# position in the alignment.

target = self.residues[stru\_id]

# It might actually be a gap; return None of it is

**if** type(target) **is** Gap:

**return** None

# Otherwise, return the id of the residue. It's a Biopython

# Residue object, so this is done with its get\_id() method.

**else**:

**return** self.residues[stru\_id].get\_id()[1]

**class** NotFoundError(Exception):

**pass**

**class** Zdiff(object):

**def** \_\_init\_\_(self, \*stru\_seq\_pairlist):

pos\_inputs = list()

**for** structure, sequence **in** stru\_seq\_pairlist:

residues = structure.get\_residues()

pairs\_for\_pos = list()

**for** letter **in** sequence:

**if** letter == '-':

seq\_unit = Gap()

**else**:

seq\_unit = residues.next()

pairs\_for\_pos.append((structure.get\_id(), seq\_unit))

pos\_inputs.append(pairs\_for\_pos)

self.positions = [Position(id\_res\_pairlist) \

**for** id\_res\_pairlist **in** zip(\*pos\_inputs)]

**def** get(self, template\_id, unknown\_id, resi, start=0):

**for** pos **in** self.positions[start:]:

**if** pos.resi(unknown\_id) == resi:

**return** pos.zdiff(template\_id, unknown\_id)

**raise** NotFoundError('resi {} of {} not found' \

.format(resi, unknown\_id))

**def** report(self, template\_id, unknown\_id):

output = (pos.zdiff(template\_id, unknown\_id) \

**for** pos **in** self.positions)

**return** filter(lambda x: x **is** **not** None, output)

**def** resi\_report(self, template\_id, unknown\_id):

output = ((pos.resi(template\_id),

pos.zdiff(template\_id, unknown\_id)) \

**for** pos **in** self.positions)

**return** filter(lambda x: x[1] **is** **not** None, output)

# The API for making zdiff files

**def** calc(template\_name, target\_name, template\_structure\_filename,

target\_structure\_filename, alignment\_filename,

write\_to, comment, format\_='clustal'):

# Open relevant files

with open(template\_structure\_filename, 'r') as template\_structure\_file,\

open(target\_structure\_filename, 'r') as target\_structure\_file,\

open(alignment\_filename, 'r') as alignment\_file:

# Load structures with Biopython's PDB file parser

# Daniel's aligned structures are missing some inessential

# information, and as a consequence the parser gives thousands

# of warnings. Gotta ignore these.

with warnings.catch\_warnings():

warnings.simplefilter('ignore')

template\_structure = PDBParser().\

get\_structure(template\_name,

template\_structure\_file)

target\_structure = PDBParser().\

get\_structure(target\_name,

target\_structure\_file)

# Open alignment using Biopython's parser

alignment = AlignIO.read(alignment\_filename, format\_)

# Find the template and target sequences in the alignment

**for** seq\_record **in** alignment:

**if** seq\_record.id == template\_name:

templ\_seq = seq\_record

**if** seq\_record.id == target\_name:

targ\_seq = seq\_record

# Calculate zdiff

results = Zdiff((template\_structure, templ\_seq),

(target\_structure, targ\_seq))

# Write results to a file

with open(write\_to, 'w') as o:

# Write some coments so I know which zdiff file this is

o.write('# Template: ')

o.write(template\_name + ' (' + template\_structure\_filename + ')\n')

o.write('# Target: ')

o.write(target\_name + ' (' + target\_structure\_filename + ')\n')

o.write('# Alignment: ' + alignment\_filename + '\n')

o.write('# ' + comment + '\n')

# Write the actual data

# Weird quirk of the zdiff objects - before giving a report, it

# requires the id's of the structures. I don't know if I wrote

# it to support more than two, or if I was planning to, or what.

**for** resi, zdiff **in** results.resi\_report(template\_name, target\_name):

o.write(str(resi) + ', ' + str(zdiff) + '\n')

I ran the "calc" function as so:

calc('aligned\_1A0S', 'aligned\_1AF6', 'aligned\_1A0S.pdb', 'aligned\_1AF6.pdb', 'Swiss-PDB structural alignment.aln', 'new.zdiff', 'reproduction of structural alignment to test code')

I copied the folder "june 2012 lab metting/pdbs from hhomp" to "zdiff module/pdbs from hhomp". This has the pdb structures of the proteins of known structure that are nearby other proteins of known structure in the HHOMP clustermap.