1. **align2d.py**

**Script**

from modeller import \*

env = environ()

aln = alignment(env)

mdl = model(env, file='1gz8', model\_segment=('FIRST:A','LAST:A'))

aln.append\_model(mdl, align\_codes='1gz8A', atom\_files='1gz8.pdb')

aln.append(file='DMI.ali', align\_codes='DMI')

aln.align2d()

aln.write(file='DMI-1gz8A.ali', alignment\_format='PIR')

aln.write(file='DMI-1gz8A.pap', alignment\_format='PAP')

1. **model-single.py**

**Script**

from modeller import \*

from modeller.automodel import \*

#from modeller import soap\_protein\_od

env = environ()

a = automodel(env, alnfile='DMI-1gz8A.ali',

knowns='1gz8A', sequence='DMI',

assess\_methods=(assess.DOPE,

#soap\_protein\_od.Scorer(),

assess.GA341))

a.starting\_model = 1

a.ending\_model = 5

a.make()

**Result**

1. **evaluate\_model.py**

**Script**

from modeller import \*

from modeller.scripts import complete\_pdb

log.verbose() # request verbose output

env = environ()

env.libs.topology.read(file='$(LIB)/top\_heav.lib') # read topology

env.libs.parameters.read(file='$(LIB)/par.lib') # read parameters

# read model file

mdl = complete\_pdb(env, 'DMI.B99990002.pdb')

# Assess with DOPE:

s = selection(mdl) # all atom selection

s.assess\_dope(output='ENERGY\_PROFILE NO\_REPORT', file='DMI.profile',

normalize\_profile=True, smoothing\_window=15)

1. **evaluate.template.py**

**Script**

from modeller import \*

from modeller.scripts import complete\_pdb

log.verbose() # request verbose output

env = environ()

env.libs.topology.read(file='$(LIB)/top\_heav.lib') # read topology

env.libs.parameters.read(file='$(LIB)/par.lib') # read parameters

# directories for input atom files

env.io.atom\_files\_directory = './:../atom\_files'

# read model file

mdl = complete\_pdb(env, '1gz8.pdb', model\_segment=('FIRST:A', 'LAST:A'))

s = selection(mdl)

s.assess\_dope(output='ENERGY\_PROFILE NO\_REPORT', file='1gz8A.profile',

normalize\_profile=True, smoothing\_window=15)

1. **plot\_profile.py**

**Script**

import pylab

import modeller

def r\_enumerate(seq):

"""Enumerate a sequence in reverse order"""

# Note that we don't use reversed() since Python 2.3 doesn't have it

num = len(seq) - 1

while num >= 0:

yield num, seq[num]

num -= 1

def get\_profile(profile\_file, seq):

"""Read `profile\_file` into a Python array, and add gaps corresponding to

the alignment sequence `seq`."""

# Read all non-comment and non-blank lines from the file:

f = open(profile\_file)

vals = []

for line in f:

if not line.startswith('#') and len(line) > 10:

spl = line.split()

vals.append(float(spl[-1]))

# Insert gaps into the profile corresponding to those in seq:

for n, res in r\_enumerate(seq.residues):

for gap in range(res.get\_leading\_gaps()):

vals.insert(n, None)

# Add a gap at position '0', so that we effectively count from 1:

vals.insert(0, None)

return vals

e = modeller.environ()

a = modeller.alignment(e, file='DMI-1gz8A.ali')

template = get\_profile('1gz8A.profile', a['1gz8A'])

model = get\_profile('DMI.profile', a['DMI'])

# Plot the template and model profiles in the same plot for comparison:

pylab.figure(1, figsize=(10,6))

pylab.xlabel('Alignment position')

pylab.ylabel('DOPE per-residue score')

pylab.plot(model, color='red', linewidth=2, label='Model')

pylab.plot(template, color='green', linewidth=2, label='Template')

pylab.legend()

pylab.savefig('dope\_profile.png', dpi=65)