# Homology Modelling of hcry1 Using Modeller

| | |
|---|---|
| Target Sequence: | hcry1.fasta |
| Protein: | Cryptochrome-1 |
| Gene: | CRY1 |
| Organism: | *Homo sapiens* |
| Function: | Transcriptional repressor |

Step 1: Prepare .ALI file of the target hcry1

Edit the target FASTA to write a "target.ALI" file.

```
>P1;target
sequence:target:1::491:::::
GVVNAVHWFRKGLRLHDNPALKECIQGADTIRCVYILDPWFAGSSNVGINRWRFLLQCLE
DLDANLRKLNSRLFVIRGQPADVFPRLFKEWNITKLSIEYDSEPFGKERDAAIKKLATEA
GVEVIVRISHTLYDLDKIIELNGGQPPLTYKRFQTLISKMEPLEIPVETITSEVIEKCTT
PLSDDHDEKYGVPSLEELGFDTDGLSSAVWPGGETEALTRLERHLERKAWVANFERPRMN
ANSLLASPTGLSPYLRFGCLSCRLFYFKLTDLYKKVKKNSSPPLSLYGQLLWREFFYTAA
TNNPRFDKMEGNPICVQIPWDKNPEALAKWAEGRTGFPWIDAIMTQLRQEGWIHHLARHA
VACFLTRGDLWISWEEGMKVFEELLLDADWSINAGSWMWLSCSSFFQQFFHCYCPVGFGR
RTDPNGDYIRRYLPVLRGFPAKYIYDPWNAPEGIQKVAKCLIGVNYPKPMVNHAEASRLN
IERMKQIYQQ*
```

Step 2: Prepare a .PIR sequence database for hcry1

Run NCBI-BLAST to pick up similar sequences of known structures and download. Convert to .PIR. The local database input file "pdb_seq_database.PIR" for Modeller is ready.

Step 3: Multiple Sequence Alignment & Selection of Templates

Run the Python script "build_profile.py" to fetch a MSA and write to output files "build_profile.prf", "build_profile.ali", "build_profile.txt"

```
from modeller import *

log.verbose()
env = Environ()

#Read the sequence database
sdb = SequenceDB(env)
sdb.read(seq_database_file='pdb_95.pir', seq_database_format='PIR', chains_list='ALL',
minmax_db_seq_len=(30, 4000), clean_sequences=True)

#Write the sequence database in binary form
sdb.write(seq_database_file='pdb_95.bin', seq_database_format='BINARY',chains_list='ALL')

#Read the binary database
sdb.read(seq_database_file='pdb_95.bin', seq_database_format='BINARY',chains_list='ALL')
```

```
#Read the target sequence
aln = Alignment(env)
aln.append(file='target.ali', alignment_format='PIR', align_codes='ALL')
#Convert the input sequence into a profile format
prf = aln.to_profile()

#Scan sequence database to pick up homologous sequences
prf.build(sdb, matrix_offset=-450, rr_file='${LIB}/blosum62.sim.mat',gap_penalties_1d=
(-500, -50), n_prof_iterations=1, check_profile=False, max_aln_evalue=0.01)

#Write the profile in text format
prf.write(file='build_profile.prf', profile_format='TEXT')

#Convert the profile to alignment format
aln = prf.to_alignment()

#Write the alignment file
aln.write(file='build_profile.ali', alignment_format='PIR')
```

Results: build_profile.prf
build_profile.ali
build_profile.txt

Step 4: Target-Template Alignment

Selected templates:

| | | | |
|---|---|---|---|
| 4K0R_A | [ Mouse Cryptochrome 1 | 2.65A] | |
| 4CT0_A | [ Mouse Cryptochrome 1 in complex with Period2 | 2.45A] | |
| 5T5X_A | [ Mouse Cryptochrome 1 | | 1.84A] |
| 6KX4_A | [ Mouse Cryptochrome 1 apoform | | 2.00A] |

Run the Python Script "align2d.py" for selected templates.

```
from modeller import *

env = Environ()

aln = Alignment(env)
mdl = Model(env, file='5T5X', model_segment=('FIRST:A','LAST:A'))
aln.append_model(mdl, align_codes='5T5X', atom_files='5T5X.pdb')
aln.append(file='target.ali', align_codes='target')
aln.align2d(max_gap_length=50)

aln.write(file='1.ali', alignment_format='PIR')
aln.write(file='1.pap', alignment_format='PAP')
```

Results: 1.ali
1.pap

Step 5: Model Building

Run the Python Script "model-single.py" to generate models

```
from modeller import *
from modeller.automodel import *
#from modeller import soap_protein_od

env = Environ()
a = AutoModel(env, alnfile='1.ali',
        knowns=5T5X', sequence='target',
        assess_methods=(assess.DOPE,
                #soap_protein_od.Scorer(),
                assess.GA341))
a.starting_model = 1
a.ending_model = 2
a.make()
```