PRÀCT2 : TROBAR PROTEINES HOMÒLOGUES AMB BLAST I HIDDEN MARKOV MODELS

Tenim una proteïna de la qual només sabem la seva seqüència, no sabem ni la estructura ni la seva funció ***(target)***. Volem saber si existeix una proteïna amb estructura coneguda que sigui homòloga a la nostra (que tingui un ancestre comú) ***(template)***. Ens interessa trobar templates perquè les farem servir per modelar l'estructura de la nostra target. Hi ha 3 estratègies principals per ***trobar templates:***

1. BLAST i PSIBLAST
2. Hidden Merkov Models (HMM)
3. Structure superimposition

Trobar proteïnes homòlogues fent servir BLAST i PSI-BLAST.

BLAST (Basic Local Alignment Search Tool): és un programa per comparar seqüències i fer local sequence alignments. El fem servir per trobar seqüències similars a la nostra target a les bases de dades de seqüències. Els alineaments que troba BLAST i que tenen un ***score*** estadísticament significatiu, es diuen **hits.** Aquests hits els podem evaluar segons el seu ***score*** (indica com de similars són les dues seqüències comparades) i l' **e-value** (indica el nombre de hits que per atzar, tenen millor score), però també té en compte altres paràmetres com la mida de la base de dades o la llargada de l'alineament. Ens interessa que el **score** sigui **alt** i l'**e-value** molt **baix**.

BLAST computa els scores fent servir **matrius de substitució**. Les matrius de substitució contenen scres associats a la freqüència de substitució de dos aminoàcids. Aquestes matrius s'obtenen de **MSA (multiple sequence alignments).** Les matrius poden ser:

- Non position-specific: els scores per les substitucions són iguals en totes les posicions de l'alineament. Es computen totes les freqüències de substitucions per a totes les posicions a la vegada.
- Position-specific: els scores per a les substitucions són diferents per a cada posició de l'alineament. Es computen les freqüències de substitucions per a cada posició del MSA per separat.

Normalment, BLAST fa servir la matiu de substitució BLOSUM62

PSI-BLAST és la versió **iterativa** de BLAST i obté unes matrius de substitució més específiques, de manera que es més eficaç trobant proteïnes homòlogues. PSI-BLAST comença utilitzant la BLOSUM62 per trobar proteïnes homòlogues. Després troba hits, alinea les seves seqüències fent un MSA i després construeix una **position-specific substitution matrix (PSSM).** A continuació utilitza aquesta PSSM per buscar nous hits i va actualitzant la PSSM. Això permet que, després de varies iteracions, la PSSM sigui específica pels tipus de substitucions que es dónen entre les proteïnes homòlogues a la nostra target, de manera que hi ha molta més **especificitat**.

Workflow psiblast:
Iteration 0 → Target + Blosum62 → [similar sequences] 0 → MSA 0 + Target → PSSM 0
Iteration 1 → PSSM 0 → [similar sequences] 1 → MSA 1 + Target → PSSM 1
Iteration 2 → PSSM 1 → [similar sequences] 2 → MSA 2 + Target → PSSM 2

Pas 1: BLAST → trobar proteïnes amb estructura coneguda homòlogues a la notra target.

blastp –query target.fa **-db** /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq **-out** target_pdb.out

Pas 2: PSI-BLAST → fem un exemple amb 5 iteracions. Cada vegada que surt per pantalla "round" sabem per quina iteració va.

psiblast -query target.fa **-db** /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq **-num_iterations** 5 **–out** target_pdb_5.out

El problema és que la base de dades de **PDB** és **redundant**. És possible que trobem la mateixa proteïna molts cops, de manera que la matriu de substitució no es fa bé, està esbiaixada. Aquesta PSSM anirà cap a aquelles famílies amb major nombre d'estructures conegudes i per tant no incorporarà dades evolutives d'insercions, delecions i substitucions. Per solucionar això hem de fer servir una base de dades **no redundant.** La PSSM resultant representarà la infornació evolutiva de la familia de la target amb major precisió. Aquesta base de dades no redundant és **SwissProt.**

psiblast -query target.fa **-num_iterations** 5 **-out_pssm** target_sprot5.pssm **-out** target_sprot_5.out **-db** /mnt/NFS_UPF/soft/databases/blastdat/uniprot_sprot.fasta

Aquí, creem una PSSM (-out_pssm) i un fitxer out amb el resultat del psi-blast.

Ara utilitzem aquesta PSSM que just hem creat, per buscar a la base de dades PDB (sabem que les seqüències de PDB tenen una estructura coneguda).

psiblast -db /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq **-in_pssm** target_sprot5.pssm **-out** target_pdb_sprot5.out

Important saber que aquí no hem de fer iteracions perquè no ens interessa fer-les quan estem utilitzant la base de dades PDB, ja que si ho fem, aquesta tornarà a estar esbiaixada.
També, fixa't que aquí no estem utilitzant la query (target) perquè el programa està incloent automàticament la informació de la target que hem fet servir per fer la PSSM que hem utilitzat per fer la cerca.

Pas 3: PSI-BLAST amb alineaments de ClustalW

Sabem que les PSSM s'obtenen de MSA. Aquest MSA es pot obtenir mitjançant PSI-BLAST o bé amb un programa de d'alineament de seqüència com **ClustalW**. ClustalW et permet seleccionar específicament les seqüències que vols que formin la PSSM. Així, pots introduir el MSA del ClustalW al PSI-BLAST i buscar una seqüència a PDB. Només has de **concatenar les seqüències** que tu vols alinear en un ***fasta file.***

Aquí has de crear un fitxer fasta que anomenem file.list amb les seqüències que t'interessin (copy-paste) de l'**output del PSI-BLAST de SwissProt.**

gedit file.list
fas copy-paste del NOM de les seqüències de l'output de PSI-BLAST que tinguin e-value interessant.

A continuació, s'utilitza **FetchFasta.pl** per obtenir la seqüència en format fasta de les seqüències interessants. Aniran a parar a un fitxer **multifasta**.

FetchFasta.pl –i file.list
-d /mnt/NFS_UPF/soft/databases/blastdat/uniprot_sprot.fasta **-o** file.fasta

Ara, en aquest multifasta (file.fasta) hem d'afegir la seqüència de la nostra target! (target.fa). És important que aquesta sigui la primera seqüència del multifasta per indicar la posició dels residus de la PSSM.

cat target.fa **>** pssm.fasta *(també es pot fer fent copy-paste directament)*
cat file.fasta **>** pssm.fasta

Ara fem el **ClustalW:**
clustalw pssm.fasta

Eliminem línies i asteriscs o qualsevol altre símbol que no sigui aminoàcid de l'output del ClustalW.

Finalment, introduïm aquest MSA com un input per PSI-BLAST amb l'opció **-in_msa**.

psiblast –in_msa clustalw.pssm **–out** target_pdb_specific.out **-db** /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq

Trobar proteïnes homòlogues fent servir Hidden Markov Models (HMM)

Un **Hidden Markov Model (HMM)** és un model que produeix aminoàcids un per un. Aquests aminoàcids estan produïts segons les probabilitats contingudes en el HMM. *(Bàsicament, es tracta d'una matriu que es basa en les probabilitats de certs aminoàcids característics d'una família determinada de proteïnes).* La probabilitat que té un HMM de produïr una seqüència determinada és el producte de totes les probabilitats individuals de produir aminoacids en l'ordre correcte.

A més a més, els HMMs tenen diferents **estats**: **emissió** (=conservat), **deleció** i **inserció**. El canvi d'estat a estat també es determinat per les probabilitats contingudes a l'HMM. Un sol HMM pot produïr la mateixa seqüència per diferents camins.

Pas 1: Construir un HMM amb hmmbuild

Volem construir una HMM d'una família de proteïnes en concret. Necessitem abans un MSA de les seqüències, aquest MSA es diu ***seed***. L'alineament ha d'estar amb **format stockholm** (alignment.sto)

hmmbuild globins4.hmm globins4.sto
globins.sto es el MSA en format stockholm de la base de dades PFAM. Conté moltes seqüències de dominis globins. L'HMM resultant donarà informació de les relacions evolutives de les globines.

Pas 2: Buscar seqüències similars amb hmmsearch

Ara busquem a PDB, seqüències que segueixin el HMM que hem creat, és a dir, aquelles que serien creades pel HMM amb gran probabilitat.

hmmsearch globins4.hmm /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq **>** globins_pdb.out
El resultat són seqüències ordenades per e-value.

A més a més, també podem seleccionar una seqüència concreta i fer un HMM per un domini determinat d'aquesta proteïna (és a dir, podem obtenir un HMM diferent per cada domini). Fent això, podem identificar regions que s'adeqüin al HMM.

hmmbuild fn3.hmm fn3.sto → crees un nou HMM per al domini fn3.
hmmsearch fn3.hmm single.prot **>** fn3.out →single prot conté la seqüència d'una proteïna. A fn3.out tindrem els residus de single.prot que formen el domini. Al final de tot del document hi ha un alineament.

Pas 3: Utilitzar bases de dades HMM (hmmpress i hmmscan)

En aquest cas, no volem construir un HMM sinó que volem utilitzar un que ja està disponible en una base de dades. Ara, creem una petita base de dades que anomenem **minifam** per a què ens serveixi d'exemple de com fer servir bases de dades HMM ja existents. Utilitzem el HMM que ja hem creat anteriorment i creem un altre:

hmmbuild fn3.hmm fn3.sto
hmmbuild Pkinase.hmm Pkinase.sto
cat globins4.hmm fn3.hmm Pkinase.hmm **>** minifam → concatenem els HMM en un file.

Per tal d'analitzar seqüències i perfils molt ràpidament, hem de comprimir i indexar la minifam:
hmmpress minifam

Ara podem buscar el millor perfil per una target:
hmmscan minifam single.prot **>** singleprot_minifam.out

Quan mirem aquest output, és possible que la nostra target segueixi diferents perfils. Això passa perque les seqüències poden tenir més d'1 domini.

Hmmscan es pot fer servir per trobar perfils HMM en **bases de dades d'HMM** que ja existeixen (com per exemple PFAM).

Pas 4: MSA amb hmmalign

Podem alinear seqüències fent servir el HMM per a que 1) puguem tenir en compte les substitucions i gaps de determinada familia, i 2) és més ràpid ja que alinea seqüències amb el perfil a la vegada.
hmmalign globins4.hmm globins45.fa **>** globins45_hmm.sto
aconvertMod2.pl –in h **–out** c **<**globins45_hmm.sto**>**globins45_hmm.clu → canviem el format de stockholm a clustal.

Pas 5: Trobar seqüències homòlogues utilitzant phmmer i jackhmmer

phmmer és l'equivalent al BLAST. **jackhmmer** és l'equivalent al PSI-BLAST.

jackhmmer hbb_human globins45.fa **>** globins_jackhmmer.out
phmmer hbb_human globins45.fa **>** globins_phmmer.out

Igual que passava amb el psi-blast, hem de vigilar de no utilitzar una base de dades redundant quan fem servir **jackhmmer**.

A més a més, **jackhmmer** ens permetrà trobar 1 HMM que s'adeqüi a **tots els dominis** d'una proteïna. Amb les altres opcions, potser que cada domini seguieixi un perfil d'HMM diferent.

Pas 6: utilitzar HMMs de la base de dades PFAM

PFAM és una base de dades russa que conté els perfils HMM de varies famílies conegudes de seqüències. Podem buscar el millor perfil HMMER per la nostra target a PFAM. Seleccionarem el millor perfil i farem ***fetch***. Després fem búsqueda d'homòlegs amb estructura coneguda a PDB.

phmmer hbb_human globins45.fa **>** globins_phmmer.out → hbb_human és el nom de la família.
hmmfetch /mnt/NFS_UPF/soft/databases/pfam-3/Pfam-A.hmm "domain_hbb" **>** domain_hbb.hmm → busques a la database el nom del domini i extreus el hmm.
hmmsearch domain_hbb.hmm /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq **>** hbb_pdb_by_HMM.out → busquem a pdb les seqüències homòlogues que segueixin el HMM.

Pas 7: Canvis de format

Hem de canviar de format per tenir l'alineament en el fomat més convenient pel que volem fer.

aconvertMod2.pl –in c **–out** f **<**alignment.aln**>**alignment.fa
fasta2sto.pl alignment.fa **>** alignment.sto
sto2fasta.pl –g alignment.sto **>** alignment.fa

PRÀCT3: STRUCTURAL SIMILARITY - STAMP

STAMP és una eina que ens permet saber com de similars són dues o més estructures 3D. Per fer això, hem de superimposar les estructures i calcular el valor **RMSD (root mean standard deviation).** El RMSD informa de la desviació dels àtoms entre les estructures: com més petit sigui el RMSD, més similars són les estructures. Recorda que la superimposició es fa a partir de la seva similaritat estructural, no a partir de la seqüència. També, cal tenir en compte que això només es pot fer si les dues estructures estan disponibles a PDB.
En l'evolució, l'estructura es conserva més que la seqüència, per tant, a la pràctica 4 veurem com podem predir una estructura 3D d'una proteïna a partir d'una estructura coneguda d'una proteïna homòloga. Seqüències similars comparteixen estructures similars.

STAMP alinea seqüències de proteïnes basant-se an l'estructura **3D**. Automàticament dóna alineaments múltiples i la millor **superimposició**. El programa es fa la seva pròpia matriu de similaritat per a cada posició en les seqüències i després utilitza un algorisme per trobar el *path* que maximitza la similaritat. Primerament, STAMP compara tots els parells d'estructures i després produeix un **dendograma** a partir del qual es podran generar alineaments i superimposicions.
A més a més del **RMSD**, STAMP també dona un **score (Sc)** que mesura la confiança en la qualitat de l'alineament.

- Scores **5.5 - 9.8** → molta similaritat, suggereix una relació funcional i/o evolutiva.
- Scores **2.5 - 5.5** → proteïnes relacionades però més distants. No sempre suggereix una relació funcional i/o evolutiva.
- Scores **<2.5** → molt poca similaritat estructural.

Per utilitzar STAMP, primer de tot s'ha de generar un ***input file.*** Aquest document conté la informació que necessita STAMP per alinear les proteïnes (normalment va bé utilitzar l'extensió **.domains**). El document té tantes línies com proteïnes volem alinear + una línia en blanc al final. Cada linia conté:

<nom del fitxer> <identificació proteïna> {objecte}

(Exemple a la dreta)

```
./pdb2hhb.ent 2hhba {CHAIN A}
./pdb2hhb.ent 2hhbb {CHAIN B}
./pdb1lh1.ent 1lh1 {ALL}
./pdb2lhb.ent 2lhb {ALL}
./pdb4mbn.ent 4mbn {ALL}
./pdb1ecd.ent 1ecd {ALL}
```

Hi ha dos tipus de STAMP:

- **Rough STAMP:** com a input només **dones les estructures de les proteïnes**. Funciona per a **proteïnes homòlogues** o **proteïnes amb llargada similar** encara que no tinguin similitud de seqüència. El programa haurà de trobar per ell mateix el millor path de N a C-terminal. En el cas que s'obtinguin scores molt baixos, s'ha de fer servir l'altre tipus de STAMP.
- **Alignfit STAMP:** proporciones, a més de les estructures, un **MSA** per a que el programa sàpiga quines parts ha de comparar i alinear.

ROUGHFIT

ROUGHFIT és el mode més simple d'STAMP ja que compara estructures només amb la informació de les coordenades.

stamp –l globins.domains **–rough –n 2 –prefix** globins **>** globins.out

-rough *vol dir que fa la superimposició inicial*
-n 2 *vol dir que el programa fa un ajust basat en la conformació per tal de corregir alguns errors de la superimposició inicial.*
globins.out *conté el dendograma amb els scores de cada cluster (proteïnes agrupades per similaritat). També podem veure els scores, RMSD, les llargades, el nombre d'àtoms alineats (**Nfit**, ha de ser molt similar per poder comparar RMSDs), el nombre de residus equivalents…*

A més a més, com a resultat d'aquest comando també es formen altres fitxers:

- **globins.mat**: conté la informació per fer l'arbre. És una matriu amb els scores de cada parell d'àtoms alineats.

- **stamp_rough.trans**: un fitxer que conté la informació necessaria per a generar coordenades superimposades.

- **globins.N**: un conjunt de fitxers que contenen transformacions i alineaments creats durant l'anàlisi de clusters fets per a totes les comparacions de parells. Cada fitxer (hi ha més d'1) correspon a un *node* del dendograma. Com més gran sigui la N, vol dir que les estructures són més diferents. Les estructures similars s'agrupen al principi. El nombre de *nodes* no el sabem abans de fer STAMP (surten de tots els clusters generats)**. El node amb major N conté totes les proteïnes.**
  Dins de **globins.5** *(suposant que 5 és la N més gran*), trobem: informació per a generar les **coordenades** superimposades, **detalls de similaritat** (scores, RMSD…) i un alineament **estructural** en format **block**.

Generar coordenades transformades: TRANSFORM

Podem utilitzar el programa **transform** per obtenir un pdb de les estructures superimposades d'un cluster en concret obtingut per STAMP. Per exmple el pdb de la superimposició de totes les estructures que estem estudiant:

transform –f globins.5 **–g –o** globins5_stamp.pdb → les opcions -**g** i -**o** posen les coordenades transformades de cada estructura en un mateix fitxer. A més, cada estructura estarà etiquetada seqüencialment amb un identificador (i.e A, B, C, D, E).

Per defecte, transform no inclou heteroàtoms. Si t'interessa incloure'ls, es pot afegir **-het** a la comanda transform. Si vols incloure aigües pots posar **-hoh.**

Ara, podem visualitzar la superimposició per a veure la similaritat:

rasmol globins5_stamp.pdb

Veure l'alineament estructural: ACONVERT

El programa **aconvert** converteix alineaments d'un format a un altre. En el nostre cas, podem passar l'alineament en format block a un que ens agradi més, com per exemple clustal.

aconvert –in b **–out** c **<**globin.5 **>** globin5_stamp.aln → important esborrar interrogants i space.

ALIGNFIT

Quan obtenim scores molt baixos utilitzant ROUGHFIT, passem a fer servir **l'ALIGNFIT**. Aquí, la solució que estem donant és proveïr un MSA per tal de guiar la superimposició. Per fer servir STAMP amb un MSA hem de seguir els següents passos:

1. Primer, **pdbseq** per extreure les seqüències en format fasta de les proteïnes que hi ha dins el fitxer **.domains:**

```
pdbseq –f s_prot.domains > s_prot.fa
```

2. Ara hem d'**editar** s_prot.fa i canviar els noms llargs del principi de cada seqüència, per l'identificador que nosaltres vulguem. Important no treure el '>'. Hem de posar noms que puguem identificar fàcilment.

3. A continuació hem de fer un **MSA** amb **clustalw:**

```
clustalw s_prot.fa
```

Com a output obtenim diversos fitxers, un dels quals és s_prot.aln, que conté el **MSA**.

4. Fem servir aconvert per passar el **MSA** de **clustal** a **block,** perquè STAMP no pot llegir clustal, només llegeix alineaments en format block.

```
aconvert -in c -out b <s_prot.aln> s_prot.block
```

5. Fem **alignfit** per incorporar la informació del MSA

```
alignfit -f s_prot.block -d s_prot.domains -out s_prot.trans
```

6. **STAMP**

```
stamp -l s_prot.trans -prefix s_prot_OK > s_prot_OK.out
```

Fins aquí hem fet el STAMP ALIGNFIT. Ara podem voler veure l'alineament estructural que ha generat STAMP, de manera que tornarem a fer servir els passos de transform i aconvert que hem fet abans.

```
transform - f s_prot_OK.3 -g -o s_prot_OK3_stamp.pdb
aconvert -in b -out c <s_prot_OK.3> s_prot_OK3_stamp.aln
```

També podem tornar a utilitzar **rasmol** per veure la superimposició.

Per tant, després de fer servir STAMP obtenim:

- 1 PDB amb la **superimposició**.
- 1 **alineament** estructural.

PRÀCT4: COMPARATIVE HOMOLOGY MODELING

En aquesta pràctica volem predir l'estructura 3D d'una proteïna de la qual no disposem d'informació estructural experimental. Això ho podem fer sempre i quan hi hagi una proteïna homòloga, de la qual si es tingui l'estructura experimental, que ens pugui guiar durant el procés. La nostra proteïna és la ***target*** mentre que les homòlogues amb estructura coneguda són les ***templates***.

L'estratègia que es segueix per a predir l'estructura 3D d'una proteïna és:

1. **Selecció de Templates:** ho fem mitjançant alineaments. Al final obtenim un alineament de seqüència i un alineament estructural de les templates. Podem utilitzar diferènts mètodes:
   a. **ClustalW:** buscar templates i alinearles directament amb ClustalW (pràctica 4.1)
   b. **HMM-PFAM:** buscar HMMs a PFAM, extreure el seu HMM i alinear les templates amb la query seguint aquest HMM. És un HMM extret de seqüència. (assignment 3)
   c. **STAMP-HMMER:** buscar templates, fer un STAMP i alinear query amb templates fent servir un HMM fet a partir de les templates. És un HMM extret d'estructura. (pràct 4.2).
2. **Construcció del model** fent servir **MODELLER** (pràct 4.3).
3. **Minimització de l'energia amb NAMD.** Depenent de la homologia entre target i templates, és possible que els nostres models tinguin "bad contacts" sobretot a les cadenes laterals (pràct 4.4)
4. **Avaluació del model: qualitat estereoquímica**, utilitzant **PROCHECK** (pràct 4.5).
5. **Avaluació del model: perfil energètic**, utilitzant **PROSA** (pràct 5.1).
6. **Avaluació del model: localització de gaps i distorsions estructurals**, utilitzant **DSSP** i **PSIPRED** (pràct 5.2)

SELECCIÓ DE TEMPLATES (pract 4.1-4.2)

Pas 1: PSI-BLAST

El primer pas es **trobar possibles templates** a una base de dades. Això ho fem utilitzant **PSI-BLAST.** Com hem vist a la pràctica 2, podem utilitzar PSI-BLAST a partir de la base de dades PDB o bé podem fer una cerca a SwissProt extreure el perfil HMM i fer una cerca amb aquella matriu a PDB.

psiblast -query P11018.fa **-db** /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq **-num_iterations** 1 **-out** P11018_pdb.out

Només **1 iteració** perquè estem fent psiblast prot contra pdb data base. Query és el fasta del target. Resultat PSI-BLAST P11018_pdb.out ls i **gedit** per mirar **l'output**.

1mee_A mol:protein length:275 MESENTERICOPEPTIDASE 226 2e-72

Aquesta és una de les templates que podem fer servir. Veiem que és el PDB 1mee i la chain A.
Triem fins a 5 templates. Copiem els fitxer fasta de les templates al nostre directori (podem fer això perquè prèviament la Núria s'ha descarregat els PDBs i ha fet PDBtoSplitChain, de manera que ha separat les seqüències de cada cadena en diferents fitxers).

cp 1meeA* ../
cp 1yjcA* ../
cp 1scjA* ../
cp 1s01A* ../
cp 1aqnA* ../

A partir del què trobem amb PSI-BLAST, haurem de triar les templates que vulguem utilitzar. Per analitzar la qualitat de les templates, a part dels scores hem de tenir en compte els següents factors:

- **La qualitat experimental de l'estructura:** si hi ha errors a una template, aquests els passarem al nostre model. S'ha de triar les templates de millor qualitat (millor ressolució, etc).
- **L'equivalència de l'ambient:** l'ambient ha de ser el més similar possible (per exemple la presència d'un lligand).
- **Similaritat filogenètica:** és millor seleccionar templates més properes evolutivament a la target.

Si una template obre gaps de manera random i a més no soluciona cap dels gaps de les altres templates és millor eliminar-la.

Pas 2: Alineament de seqüència

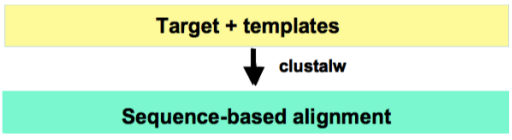
Ara, com hem dit, hi ha diferents estratègies:

CLUSTALW (pràct 4.1)

Primer hem de posar totes les templates en format fasta en un mateix fitxer **multifasta** i llavors fem el **clustalw**.

cat *fa > P11018_templates.fa

clustalw P11018_templates.fa



Tenim un alineament de seqüència amb extensió .aln.
→ P11018_templates.aln

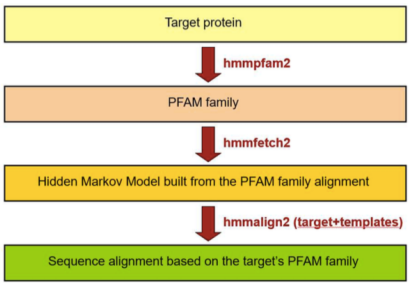
HMM-PFAM (assignment 3)

Busquem l'HMM de la família utilitzant la base de dades PFAM.

hmmpfam2 /mnt/NFS_UPF/soft/databases/pfam-2/Pfam_ls P11018.fa **>** P11018_pfam.out
gedit P11018_pfam.out

Extraiem el HMM (la matriu que seguirem per fer l'alineament). Peptidase_S8 és el nom de la família HMM que estem buscant per extreure el seu HMM. El nom surt a P11018_pfam.out del pas anterior:

hmmfetch2 /mnt/NFS_UPF/soft/databases/pfam-2/Pfam_ls Peptidase_S8 **>** Peptidase_S8.hmm
gedit Peptidase_S8.hmm

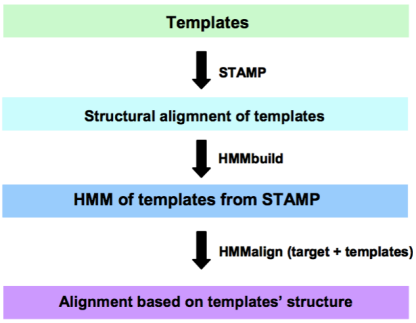


Aliniem les seqüències de les templates i la target fent servir el HMM que hem extret:

hmmalign2 Peptidase_S8.hmm P11018_templates.fa **>** P11018_pepS8.sto
gedit P11018_pepS8.sto

STAMP+HMMER - Alineament basat en l'alineament estructural de les templates (pràct 4.2)

Aquesta estrategia és més complexa perquè combina STAMP i HMMER, ja que és un alineament guiat per la informació estructural de les templates. Haurem de fer un **STAMP** i crear un perfil **HMM** a partir de la informació de l'**alineament estructural.**



Primer hem de crear l'**input file**: templates.domains. Per no escriure massa fent un comando podem copiar els PDB ID de les nostres templates a l'input file:

ls *pdb > templates.domains

Hem d'acabar de **completar** l'arxiu perquè ens quedi així:

./1aqnA.pdb 1aqnA {all}
./1meeA.pdb 1meeA {all}
./1s01A.pdb 1s01A {all}
./1scjA.pdb 1scjA {all}
./1yjcA.pdb 1yjcA {all}

Posem *all* perquè no hem d'especificar la cadena, prèviament ja havíem fet pdbs separats per cada cadena de la proteïna. A més, per la mateixa raó, en aquest document no hi ha columna de chain.

Fem un **STAMP** i mirem el pdb amb **rasmol**:

stamp -l templates.domains **-rough -n 2 -prefix** templates **>** templates.out
transform -f templates.4 **-g -o** templates4_stamp.pdb
rasmol templates4_stamp.pdb

Obrim templates4_stamp.aln i **esborrem** l'espai i els interrogants. A continuació **construïm l'HMM.**

hmmbuild2 templates.hmm templates4_stamp.aln
hmmcalibrate2 templates.hmm

Finalment, **aliniem** les seqüències de les templates (i també la nostra target) seguint l'HMM que just acabem de crear:

hmmalign2 templates.hmm P11018_templates.fa **>** P11018_templates.sto

El que hem fet és construir el nostre **HMM** a partir de **dades estructurals** (superimposició d'STAMP - alineament estructural).

Ara, **repetim** tot el procés però en comptes d'utilitzar les templates, **utilitzem totes les subtilases (33) per fer l'HMM**. D'aquesta manera esperem obternir un HMM amb **més valor estadístic.**

stamp -l subtilases.domains **-rough -n 2 -prefix** subtilases **>** subtilases.out
transform -f subtilases.33 **-g -o** subtilases33_stamp.pdb
rasmol subtilases33_stamp.pdb
Obrir el aln i treure espai i interrogants.
hmmbuild2 subtilases.hmm t subtilases33_stamp.aln
hmmcalibrate2 subtilases.hmm
hmmalign2 subtilases.hmm P11018_templates.fa **>** P11018_ subtilases.sto

Tot i això, si observem aquest alineament, veiem que l'HMM no canvia molt. Això indica el poder de l'estructura: podem obtenir bons HMMs només utilitzant 5 templates.

BONUS: Obtenir els PDBs de les diferents cadenes de les templates que ens interessen:

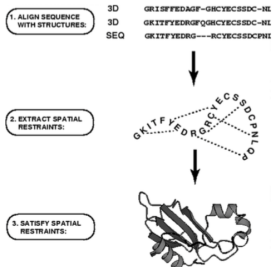
El més fàcil és anar a la web del PDB i descarregar-se les estructures de les proteïnes templates que ens interessen. Quan ens descarreguem un pdb, normalment tenim totes les cadenes formant tota l'estructura. Podem obtenir la seqüència i les coordenades de les nostres templates fent servir un programa Perl que es diu **PDBtoSplitChain.pl.** Aquest programa divideix l'arxiu PDB en tants arxius com cadenes i produeix un **fasta** i un **pdb** per a cadascuna d'elles.

PDBtoSplitChain.pl -i pdb_file **-o** prefix → com a prefix posarem el **pdb ID**.

Com a resultat d'aquesta comanda es creen arxius anomenats per exemple (pdb IDs són exemples):

1meeC.fa i 1meeC.pdb → indica que el pdb ID és 1mee, la cadena és la C i és un fasta o un pdb.

CONSTRUCCIÓ DEL MODEL - MODELLER (pràct 4.3)



Ara fem servir el programa **MODELLER** per construir el nostre model. Modeller és una implementació automàtica d'una aproximació de **modelatge comparatiu** que segueix **restriccions espacials.** Funciona de tal manera que primer, les **estructures 3D** de les templates **s'alinien** amb la target, després s'apliquen les **restriccions espacials** (distància entre Cα, Hbonds, side-chains, angles…) que es transfereixen de les templates a la target, finalment, **s'obté el model 3D**. Aquest complirà el màxim de restriccions el millor possible.

Per fer els models, es recomana **crear nous directoris**: un pels models que obtindrem a partir de l'**alineament de seqüència** i un altre pels models que obtindrem a partir de l'alineament basat en l'**alineament estructural** de les templates (**hmm**). A cadascun d'aquests directoris hem de ficar una **còpia de l'alineament** que farem servir, els **pdb de les templates** i **l'input file de modeller.**

Pas 1: Convertir l'alineament a format pir

Modeller només llegirà l'alineament si aquest està en **format pir.** El programa **aconvert** ens permet fer aquest canvi de format. L'**output** és un alineament amb tants blocs com seqüències i cada bloc té un **encapçalament** de dues línies amb **etiquetes**. Aquestes etiquetes hauran de ser les **mateixes** que les que posarem a l'**input file.**

aconvertMod2.pl -in c **-out** p **<** P11018_templates.aln **>** P11018_templates.pir

Pas 2: modificar l'input file (extensió .py)

L'**input file** conté els paràmetres que necessita Modeller i les opcions que volem fer servir per construir el nostre model 3D. Aquest document hauria de tenir l'extensió **.py.** Coses a modificar:

- **alnfile:** hem de posar el nom del nostre alineament en format pir, a partir del qual construirem els models. Va entre cometes individuals. → 'P11018_templates.pir'
- **knowns:** aquí hem de posar les etiquetes de cada template. Aquestes han de ser les mateixes que hi ha a l'alineament .pir. Etiquetes van entre cometes individuals i separades per coma i espai. → ('1meeA', '1aqnA', …)
- **sequence:** hem de posar l'etiqueta de la nostra target igual que apareix a l'alineament → 'P11018'
- Per defecte, els **pdb** han d'estar al **directori** on estem treballant. (Si no, hem de donar la seva localització).
- **a.ending_model:** posem un 2 perquè volem fer dos models. Podriem fer tants models com vulguessim perquè hi ha més d'una solució per a cada restricció espacial.
- **a.make():** no ho hem de canviar. Serveix per fer homology modeling de la manera més simple.

Un cop hem modificat l'input file, li **canviem el nom i el guardem**. Li posem de nom P11018_templates.py.

Pas 3: Run modeller

mod9.21 P11018_templates.py

Com hem dit al programa que voliem dos models, hem obtingut 2 pdbs: un amb cada model: P11018.B99990001.pdb i P11018.B99990002.pdb. Per a la nostra comoditat i per evitar sobreescriure si tornem a fer córrer el programa, els canviem el nom:

mv P11018.B99990001.pdb model1.pdb
mv P11018.B99990002.pdb. model2.pdb

Un cop tenim els models, pot ser que necessitem refinar els alineaments perquè ens ha passat el següent:

- Hi ha un **loop que trenca una estructura secundària:** indica que l'alineament ha obert un gap sense tenir en compte l'estructura secundària. Això passa sovint quen ho fem a partir de l'alineament de seqüència clustalw.
- Hi ha **segments desordenats al N-terminal o al C-terminal:** això és perquè hi ha gaps al principi i/o al final de l'alineament. És recomanable construir nous models sense aquest segment, esborrrant certs aminoacids de l'arxiu fasta. Assegura't de canviar el nom dels fasta si ho fas.
- La presència de **loops amb diferent conformació** indica la presència de gaps en el nostre alineament. (Aquestes zones s'analitzen a les pràct 4.5, 5.1 i 5.3).

Pas 4: modificar els fasta per millorar els models

Els models que hem obtingut no ens acaben d'agradar perquè hi ha segments amb gaps al principi i al final. Necessitem fer nous models que no continguin aquests gaps.

El que fem és **modificar el fasta de les templates** (P11018_templates.fa) i **esborrem** els aminoàcids que corresponen als **gaps**. Ho guardem com a un nou document: P11018_templates_mod.fa.

Ara, tornem a fer **TOT** el procés de construir el model a partir d'aquest nou fasta:

clustalw P11018_templates_mod.fa → obtenim P11018_templates_mod.aln
aconvertMod2.pl -in c **-out** p **<**P11018_templates_mod.aln**>** P11018_templates_mod.pir
gedit P11018_templates.py → Modifiquem l'input file i hem de canviar alnfile (per posar l'alineament que farem servir ara).
mod9.21 P11018_templates_mod.py
mv P11018.B99990001.pdb model_mod_1.pdb
mv P11018.B99990002.pdb. model_mod_2.pdb

Pas 5: repetir per obtenir models a partir dels hmm

hmmalign2 subtilases.hmm P11018_templates_mod.fa **>** hP11018_templates.sto
aconvertMod2.pl -in h **-out** p **<**hP11018_templates.sto **>** hP11018_templates.pir
gedit P11018_templates.py → modificar el aln file, hP11018_templates
mod9.21 P11018_templates.py
mv P11018.B99990001.pdb hmodel1.pdb
mv P11018.B99990002.pdb. hmodel2.pdb

Al final de fer tot això tenim **6 models** en total:

model1.pdb, model2.pdb → tenen la "cua".
model_mod_1.pdb, model_mod_2.pdb → no tenen la cua
hmodel1.pdb, hmodel2.pdb → no tenen la cua i s'han fet a partir de HMM.

Els següents passos, s'han de continuar amb els 4 models que **no** tenen cua.

MINIMITZACIÓ D'ENERGIA AMB NAMD (pràct 4.4 = semi3)

Com que hem fet el nostre model basant-nos en l'**estructura de les templates,** no hem tingut en compte les **cadenes laterals** (ens hem guiat pel backbone). Ara, al posar-les, és possible que aquestes estiguin massa juntes, per la qual cosa hem d'**opimitzar l'energia.** Ens interessa que el model tingui la **mínima energia possible**. L'energia augmenta si les cadenes laterals estan massa juntes.

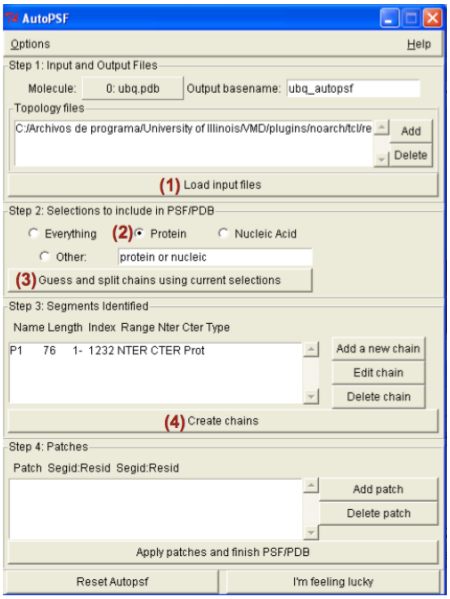
NAMD és una eina de ***molecular dynamics*** que fa **simulacions** per a biologia estructural. NAMD necessita diversos arxius:

- **PDB file:** conté les coordenades dels àtoms de l'estructura.
- **PSF (Protein structure file):** conté la informació estructural com els enllaços, els angles …
- **Force field parameter file:** conté la informació de referència del tipus d'àtoms, càrregues, com els àtoms estan connectats… Trobem coses com: *bond energy, angle energy, dihedral* (conformació de la *side chain*), *non-bonding energ*y (electrostatic/VDW).
- **Configuration file:** conté les opcions per a dur a terme la simulació.

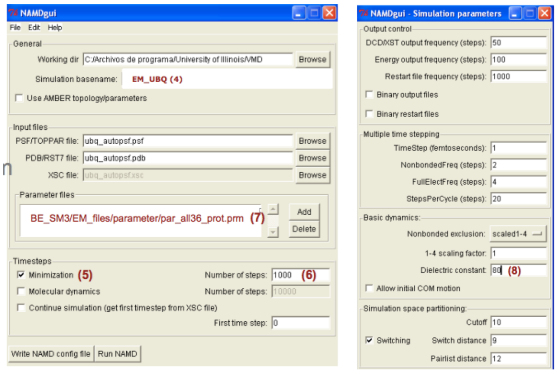
Amb això podriem pensar que cada característica mencionada aquí és independent però no és així, una sola **variació** d'un paràmetre pot afectar tots els altres.

Pas 1: Preparació del PSF i el PDB input file

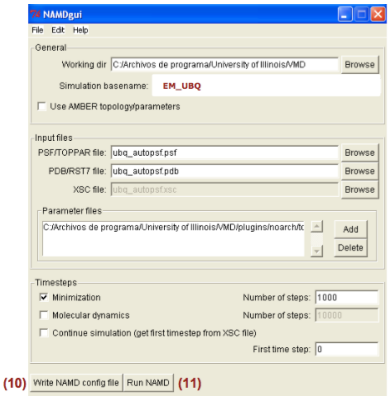
1. Obre Terminal
2. Obre VMD
3. Obre TK console desde VMD main → Extensions → TK console
4. Navega a fins al fitxer on vulguis guardar tots els teus càlculs.
5. **Generació del PSF i el PDB**
    1. Load hmodel1.pdb a VMD
    2. Obre Automatic PSF Builder (Extensions→ Modeling→ Automatic PSF Builder)
    3. Clica a **Load Input Files**
    4. Selecciona **protein**
    5. Clica a **Guess and split chains**
    6. Clica a **Create Chains**
6. Obtenim els arxius: hmodel1_autopsf.pdb i hmodel1_autopsf.psf



Pas 2: Energy minimization

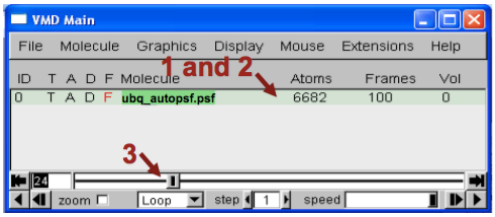


1. Esborra tot
2. Load a VMD hmodel1_autopsf.psf i hmodel1_autopsf.pdb.
3. Obre **NAMD graphical interface** (Extensions→ Simulation→ NAMD graphical interface)
4. Canvia **Simulation Base Name** (hmodel1)
5. Selecciona **Minimization**
6. Canvia el **number of steps** a **2000**
7. Ajusta el ***path*** per trobar els ***parameter files***
8. Ajusta els **Simulation Parameters** (edit→ Simulation Parameters) i posa la **constant dielèctrica** a **80** (per simular l'ambient amb aigua).
9. Clica a **Write NAMD config file**
10. Obre hmodel1.namd i assegura't que els paràmetres estan bé.
11. Clica a **Run NAMD**
12. Mira els arxius output.



Pas 3: Anàlisi del que hem obtingut

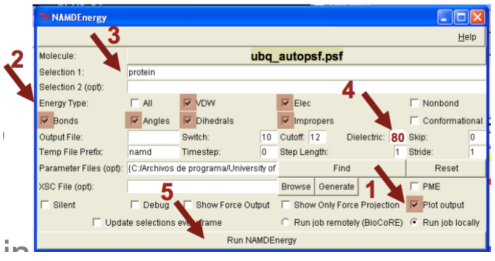
Mirem el trajectory file (**.dcd**)



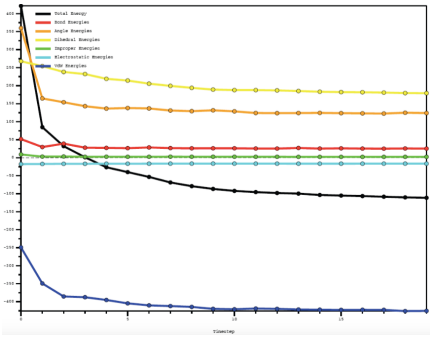
1. Load l'**estructura de la proteïna** (segurament és hmodel1_auto.psf)
2. Load el **trajectory file** (segurament hmodel1.dcd) dins de hmodel1_auto.psf.
3. Mou la barra de baix per observar el moviment que ha fet la teva estructura durant la minimització.
4. Identifica els residus que han patit un major canvi conformacional durant l'optimització.

Pas 4: Anàlisi de l'energia total del sistema

Obre **NAMD Energy**: extensions→ analysis→ NAMD Energy.



1. Selecciona **Plot output**
2. A Energy Type, selecciona: **Bonds, Angles, VDW, Dihedrals, Elec, Impropers.**
3. A Selection 1, escriu: **protein.**
4. Posa Dielectric a **80**.
5. Clica a **Run NAMDEnergy**
6. Observa el gràfic de les energies.



AVALUACIÓ DEL MODEL: QUALITAT ESTEREOQUÍMICA - PROCHECK (Pràct 4.5)

Aquí comencem a avaluar els models que hem obtingut per poder triar quin de tots és el milor , saber quines zones són més fiables i quines contenen el major nombre d'errors. Per analitzar la **qualitat estereoquímica** fem servir el programa **PROCHECK**.

Aquest programa compara l'estructura de la proteïna que nosaltres li donem (el nostre model) amb estructures "ben refinades" de la mateixa **ressolució**, i dóna una indicació de la **fiabilitat de cada residu.** Per fer això, utiitza uns paràmetres que són bons indicadors de la qualitat estereoquímica.

La **estereoquímica** és la part de la química que estudia la geometria de les molècules. La distància dels enllaços i els seus angles tendeixen a estar en equilibri (valors estadístcs). Si els enllaços del nostre model són més llargs o més curts farà que aquesta estereoquímica canviï. El que hem d'analitzar curosament són els ***dihedrals***: el plegament de la proteïna depèn dels ***dihedrals*** del ***backbone***. Depenent del valor dels dihedrals tindrem una hèlix, una ß-sheet, etc… Què es mira:

- **Angle Omega:** es forma entre un COOH i el següent NH. Aquest sempre és pla (perquè és un doble enllaç parcial). Només hi ha dos possibles posicions: **trans** (la posició normal) i **cis** (la posició més inestable). A la posició **cis** només hi pot participar la **prolina**, però no totes les prolines estan en cis. Si tens una prolina cis a la teva template però no al teu model, és molt probable que el teu model estigui malament.

- **Angle Psi:** és l'angle entre el Cα i el COOH. És un enllaç simple, així que pot rotar, però no ho fa per la presència de les cadenes laterals. Mirarem si el **Ramachandran plot** del nostre model és similar al de les templates.

Cada vegada que fem servir PROCHECK hem de crear un nou directori amb tot allò que volem analitzar (per no liar-nos).
A més, per fer servir el programa hem de tenir en compte la ressolució de les nostres templates. La ressolució és important per avaluar els dihedrals, ja que és molt important la posició dels àtoms. Així, **mirem la ressolució de les nostres templates** (ho busquem a la web de PDB).

La millor ressolució (**alta ressolució**) és aquella amb el nombre més **baix** d'angstroms. Contràriament, un nombre més alt d'angstroms significa pitjor ressolució (baixa ressolució) .

1aqn → 1.8 Å
1mee → 2 Å
1s01 → 1.7 Å
1yjc → 1.8 Å
1scj → 2 Å

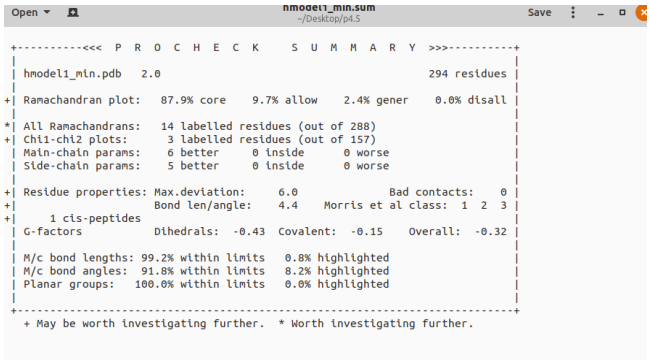
Per fer servir procheck amb la nostra estructura minimitzada (hmodel1_min.pdb), hem d'introduir el nombre de la ressolució més baixa (la pitjor, és a dir el nombre més alt).

procheck hmodel1_min.pdb 2.0

Com a resultat d'aquesta comanda obtenim molts arxius:
- hmodel1_min.sum: és un summary de les característiques de l'anàlisi.
- 10 postscript files, amb gràfics dels diferents aspectes de l'anàlisi:
  - hmodel1_min_01.ps: Ramachandran plot
  - hmodel1_min_02.ps: Ramachandran plot per gly i pro
  - hmodel1_min_03.ps: chi1-chi2 plots
  - hmodel1_min_04.ps: main-chain parameters
  - hmodel1_min_05.ps: side-chain parameters
  - hmodel1_min_06.ps: residue properties
  - hmodel1_min_07.ps: main-chain bond length distribution
  - hmodel1_min_08.ps: main-chain bond angle distribution
  - hmodel1_min_09.ps: RMS distances for planarity
  - hmodel1_min_10.ps: disorted geometry plot

- hmodel1_min.out: llista residu per residu
- Arxius amb extensions .lan, .nb, .new, .pln, .rin, .sco, .sdh són per ús intern del programa.
- 7 log files útils si falla PROCHECK o si dona resultats estranys.

El **summary** ens dóna informació sobre els % de residus en cada àrea del Ramachandran Plot, formen dos grups: els que estan en conformació **favorable** (core+allow) i els que estan en conformació **no favorable** (gener +disall). A més a més, aquest arxiu també també ens informa del nombre de ***bad contacts.*** Hem de procurar tenir el menor nombre de bad contacts possible. Normalment, si **optimitzem** com hem fet a la pràctica anterior, aquest número serà **0**. Si analitzéssim els models abans i després de la optimització potser aquest nombre canviaria.

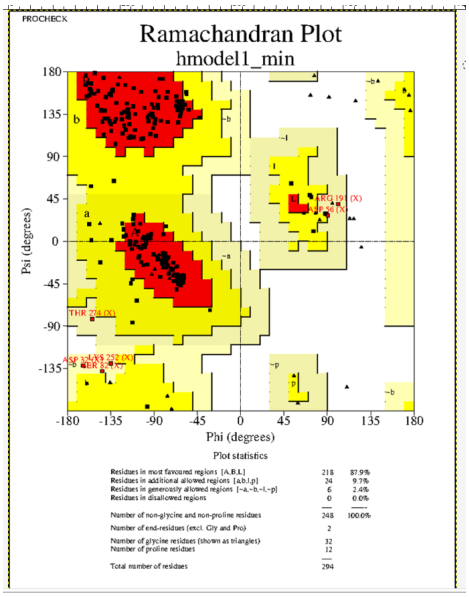


També són interessants els **cis-peptides:** en el nostre model en tenim 1. Hauriem de comprovar que les templates també el tenen, si no el tenen, podeu panicar.

Per veure i analitzar el **Ramachandran plot** utilitzem el programa **gimp:**

gimp hmodel1_min_01.ps

S'ens mostra un gràfic en què l'eix de les **x** conté els valors dels **angles phi** (-180 a 180). L'eix de les **y** conté els valors dels **angles psi** de tots els residus de la proteïna. Cada punt negre correspon a un residu de la proteïna. Aquests punts es troben dins d'àrees de diferents colors. Cada color té un significat:



- **Àrees vermelles** = ***Core***. Combinació de psi i phi amb valor negatiu (molt estable). Correspon a estructures com beta sheets i α-helix.
- **Àrees grogues** = ***Allowed***. L'energia és encara negativa. Significa que les combinacions són estables pero no són hèlix o beta sheet, són loops estables.
- **Àrees beige** = ***Generously Allowed***. Són loops que no són estables.
- **Àrees blanques** = ***Disallowed***. Són combinacions de psi i phi amb alta energia positiva (molt poc estable, està prohibit).
- **Residus amb etiquetes.** No són comuns. Són residus que estan al límit estadístic. No sabem si són per error nostre o no, perquè hem fet el model a partir de les templates. Si efectivament són un error, arrossegarem aquest error durant tot el procés. Podem mirar el plot de les templates: si no apareixen els residus amb etiquetes, vol dir que si que són culpa nostra.

PRÀCT5: AVALUACIÓ DEL MODEL

PERFIL ENERGÈTIC - PROSA (pràct 5.1)

Per predir bé l'estructura d'una proteïna és crucial que seleccionem el model més proper a la conformació nativa de la proteïna. Hi ha diversos mètodes que s'han desenvolupat per puntuar aquests models segons energies, *knowledge-based potentials* i combinació dels dos. En aquest cas fem servir **PROSA**. PROSA utilitza els scores per mirar la **qualitat d'una estructura** i comprovar (mitjançant threading) **quina és la millor** conformació d'entre tots els models. PROSA s'ha de fer abans de la optimització.

PROSA es fixa en 3 tipus d'energia:
- ***Surface***: l'energia pe que respecta la interacció amb el solvent.
- ***Pair:*** depèn de la interacció entre els aminoàcids.
- ***Combined:*** combina les dues anteriors.

Per obrir PROSA només hem d'escriure **prosa** al terminal. A partir d'aquí, tots els commandos s'escriuen a la pantalla de PROSA. La pràctica està organitzada en *sample sessions* que venen donades per PROSA i cadascuna d'elles serveix per coses diferents. Es pot cridar la sessió directament (**execute session1.cmd)** o es pot escriure tot el que conté la sessió. S'ha de tenir en compte que les sessions predeterminades tenen pdbs concrets, si volem fer servir els nostres s'han de canviar.

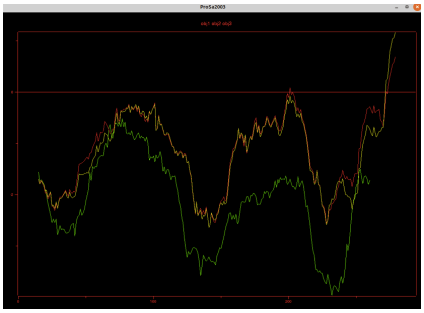
Session 1 → Llegir PDBs per calcular i visualitzar energies.

read pdb pdb2aat.ent **obj1**
analyse energy obj1
plot
winsize obj1 50 → serveix per fer un punt cada 50 residus (mitja) (el gràfic és més *smooth*)
plot
draw * obj1 1 → * = tot; 1=fes-ho.
plot
color comb obj1 cyan → canviar de color les energies per poder-les diferenciar al gràfic
color surf obj1 magenta
plot
read pdb pdb1spa.ent **obj2**
analyse energy obj2
plot
color * obj2 red
winsize * 50
plot
draw * * 0 → 0=no ho facis
draw pair * 1
plot
export plot myplot

Quan observem el *plot,* el més ideal seria que totes les energies fossin negatives → més estable.

De vegades els aminoàcids de diferents templates no acaben de quadrar, encara que mirant al gràfic tu veus que segueixen el mateix dibuix. És possible que hagis de fer un s***hift*** en alguna estructura per a que quadri. Si fem **shift 1** estem movent el gràfic 1 aminoàcid.

Comparem dos dels nostres models amb una template



Ara anem a comparar dos dels nostres models hmodel1.pdb i hmodel2.pdb amb una de les nostres templates, 1aqnA.pdb. Creem nosaltres una sessió fent un .txt que al final guardarem amb extensió .cmd. Al final fem a PROSA **execute session.cmd** que conté:

read pdb hmodel1.pdb **obj1**
read pdb hmodel2.pdb **obj2**
read pdb 1aqnA.pdb **obj3**
analyse energy * → analitza l'energia de tots els models
winsize * 30 → fa un winsize de 30 per a tots els models
draw * * 0 → NO em dibuixis (0) totes les energies de tots els models.
draw pair * 1 → dibuixa l'energia pair per a tots els models
color * obj2 red
color * obj3 green
plot
export plot myplot

Calcular els z-score

Per comprovar els z-score de cada energia i veure que està dins de la normalitat i que per tant és un bon model hem de crear una nova sessió per executar a PROSA: li podem posar de nom sessionZ.cmd i aquesta conté:

read pdb hmodel1.pdb **hmod1**
read pdb hmodel2.pdb **hmod2**
read pdb 1aqnA.pdb **1aqnA**
init zscore
combine type sdev → perquè et surtin totes les energies
zscore * zscore → calcula els zscores i els fica en un nou document que es diu zscore.slp, hem de mirar aquest!

Si obrim zscore.slp, veiem un document en forma de taula on tenim la llargada de les proteïnes i els zscores que hem obtingut (i més coses). Hem d'agafar els zscores per a cada energia i, tenint en compte la llargada de les proteïnes, posar-los en aquest gràfic i veure si es mantenen en la zona de normalitat. Sí es així, els models estan més o menys bé.

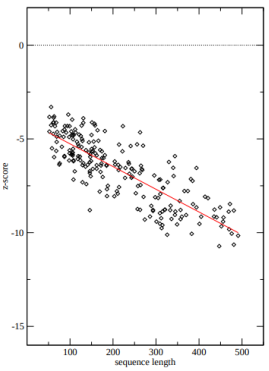


Figure 1: Z-scores of pair energy(C^β interactions). Regression equation:$y = -4.05 - 0.0121x$

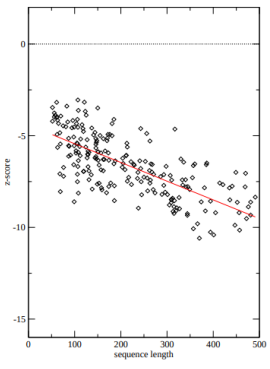


Figure 2: Z-scores of surface energy(C^β). Regression equation: $y = -4.43 - 0.0101x$

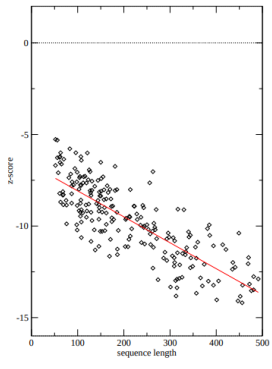


Figure 3: Z-scores of combined energy(C^β). Regression equation: $y = -6.67 - 0.0141x$

LOCALITZACIÓ DE GAPS I DISTORSIONS ESTRUCTURALS - PSIPRED i DSSP (pràct 5.2)

Hem obtingut un model que té una regió mal modelada. La única part ue està malament de tota l'estructura és aquesta part, la resta està ben modelada. En aquesta pràctica, identificarem aquesta regió mal modelada i sabrem com **corregir**-la. Per a fer això utilitzarem p**otencials estadístics** i programes per a la **predicció i identificació d'estructures secundàries**. Cal saber, però, que una altra manera de corregir el model és simplement utilitzar una altra template.

Sabem que alguns aminoàcids tendeixen a formar part d'estructures secundàries concretes. Podem utilitzar aquest coneicement per avaluar l'assignació de l'estructura secundària en un model determinat. Si trobem que algunes regions de la proteïna tenen una estrctura secundària que difereix de la predicció, és probable que aquesta regió estigui mal modelada.

A continuació utilitzarem dos programes: PSIPRED i DSSP. **PSIPRED** és un programa que **prediu** l'estructura secundària d'una proteïna a partir d'una **seqüència** en format fasta. **DSSP** és un programa que **extrau** l'estructura secundària a partir d'un **PDB**. Et diu per a cada aminoàcid, en quina estructura secundària es troba. En definitiva el que farem és **comparar** l'estructura secundària **extreta** del nostre **model** i la que **hauria de ser** segons la **predicció** de psipred feta a partir de la seqüència que té el nostre model.
Per dur a terme la pràctica, hem d'agafar el model predit abans d'optimitzar-lo.

Pas 1: comparar l'estructura secundària predita (PSIPRED) amb la modelada (DSSP)

1. Extreure la seqüència fasta del model PDB.
   PDBtoSplitChain.pl -i model1.pdb **-o** model

   Obtenim una seqüència **fasta** model.fa → la farem servir per que psipred pugui fer la seva predicció.

2. Executar **PSIPRED** amb la seqüència fasta que acabem d'extreure. Aquí estem fent la **predicció** de l'estructura.
   psipred model.fa

   Obtenim dos arxius diferents:
   - model.ss2: conté l'assignació de l'estructura secundària per a cada residu de la seqüència.
   - model.horiz: conté els scores associats a les prediccións de cada residu. Com més alt sigui l'score, més fiable és la predicció (el millor score que trobem és 9).

3. Transformar model.ss2 en un alineament en format pir que contingui la seqüència del model i l'estructura secundària.
   psipred.pl model.ss2 **>** psipred.pir

4. Extreure l'estructura secundària del model amb DSSP. Aquí estem **extraient** literalment l'estructura secundària del nostre model.
   dssp model.pdb model.dssp

5. Transformar l'output de **DSSP** a un alineament en format pir que contingui la seqüència del model i la seva estructura secundària.
   aliss.pl model.dssp **>** dssp.pir

6. Concatenar els dos alineaments pir que hem creat (el de psipred i el de dssp) en un sol arxiu pir.
   cat psipred.pir dssp.pir **>** compare.pir

7. Transformar l'arxiu concatenat pir a un alineament en format clustalw:
   aconvertMod2.pl -in p **-out** c **<**compare.pir**>**compare.clu

   Ara tenim un alineament que conté: la seqüència del model, la seva estructura secundària (DSSP) i l'estructura secundària predita (PSIPRED). Per tant, podem mirar les regions en què l'estructura predita i la del nostre model no coincideixen.
   Hèlix → H, G, I
   Strand → B, E
   Loop/coil→ C,T,S o qualsevol altra lletra no mencionada.

8. Ara, pas important, hem d'agafar la **línia dels scores** de model.horiz i **enganxar-la** a sobre de tot de compare.clu. D'aquesta manera podem determinar les zones que estan molt ben predites i modelades i les que no coincideixen gens.
   Errors comuns que podem trobar és que tenim un loop on hauriem de tenir una hèlix. Modeller fa això molt sovint perque quan no té cap informació estructural (té un gap de templates) hi fica un loop. De vegades, fent això, Modeller està trencant hèlix, cosa que no és bona.

Pas 2: Inspeccionar el model amb CHIMERA

Fins ara hem detectat errors tant amb PROSA com amb PSIPRED i DSSP. Ara mirarem el model amb CHIMERA.

1. Obrir el **model** i la **template** amb Chimera:
   chimera model.pdb template.pdb → *(template.pdb en el nostre cas seria per exemple 1aqnA.pdb)*

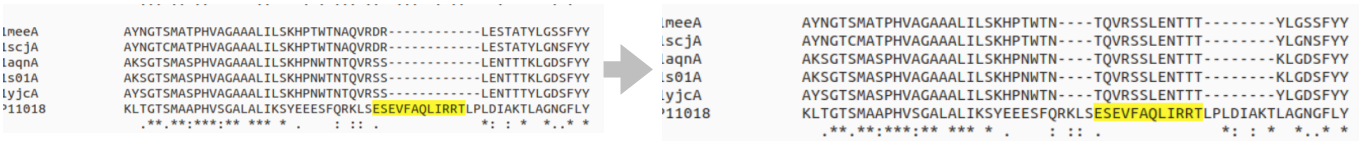
2. **Superimposar** les dues estructures utilitzant el "Match Maker". Utilitza l'eina "Sequence" per identificar la regió mal modelada (Tools → Sequence).
   **A l'examen hauriem de fer el model a partir d'1 template per tal de poder fer aquesta comparació senzilla (si utilitzem moltes hauriem de fer Chimera amb totes elles). L'única situació en que té sentit fer servir mes d'1 template és si la proteïna té diferents dominis i fas servir una template per cada domini.*

3. En **observar** les estructures en Chimera, és molt probable que veiem que el nostre model difereix de la template. Per exemple, és possible que una hèlix s'hagi trencat i tinguem un loop en mig de la hèlix. Si modifiquem alguns documents, això ho podem arreglar.

Pas 3: Corregir el model modificant l'alineament input de MODELLER.

Per corregir l'error de l'hèlix haurem de fer un nou model amb l'alineament d'input de Modeller modificat.
Per arreglar els loops extras, podem esborrar manualment el gap de la seqüència i posar-lo al final de l'hèlix. Això farà que l'hèlix estigui envoltada per loops més grans que els que esperariem, però d'aquesta manera estem conservant l'estructura del nostre model.

1. **Modificar** l'arxiu **clustalw** que hem donat com a input a **Modeller** (P11018_templates.aln):
   La regió en groc indica la seqüència que hauria de ser hèlix però que és un loop en el nostre model (ho hem vist a Chimera). Hem de fer que aquesta regió estigui alineada amb residus i així modeller ho predirà com una hèlix. Realment no estem modificant la seqüència, ja que només canviem de lloc els gaps: en comptes de tenir un gap a la meitat de l'hèlix tindrem el loop dividit al principi i al final d'aquesta.



2. Ara hem de fer **nous models** que incloguin aquest canvi:

   aconvertMod2.pl -in c **-out** p **<**P11018_templates.aln**>** target_template.pir
   modpy.sh python2 P11018_templates.py
   mv P11018.B99990001.pdb corrected_model1.pdb
   mv P11018.B99990002.pdb corrected_model2.pdb
   **No oblideu modificar P11018_templates.py si és necessari*

3. Mirem els nous models (corrected_model1.pdb i corrected_model2.pdb) a **Chimera**, superposant-los amb una template i mirem que efectivament, l'hèlix ja no està partida.

Pas 4: Comprovar la qualitat del nostre model amb PROSA.

Podem fer PROSA per mirar la qualitat del model (mirar que les energies siguin negatives com vam fer en la pràctica anterior). Podem comparar el model corregit, el model d'abans i una template.

Escriu **prosa** al terminal i dins de prosa escriu els comandos:

read pdb corrected_model.pdb **obj1**
read pdb model.pdb **obj2**
read pdb 1meeA.pdb **obj3** → 1 de les templates que has fet servir.
analyse energy *
winsize * 30
color * obj2 red
color * obj3 cyan
shift obj3 20
plot
**Potser paràmetres com el shift s'han d'ajustar! Mirar si ho necessites.*

Pas 5: (No el vam fer) Triar altres templates utilitzant HMM.

Aquest pas eren deures que no vaig fer. Es tracta de canviar les templates amb les que hem fet el model amb altres extretes mitjançant HMM. Això ho hem fet a l'*assignment 3* (crec), però aquí fan servir commandos una mica diferents. Els deixo aquí escrits tal qual surten al guió de pràctiques però jo crec que es podria fer si féssim **models** a partir de l'**alineament** que vam obtenir amb *l'assignment 3.*

1. Trobar el millor **HMM** de la **PFAM** de la nostra target:
   hmmscan /mnt/NFS_UPF/soft/databases/pfam-3/Pfam-A.hmm P11018.fa **>** hmmscan.out

2. Fetch l'HMM que hem trobat a la base de dades de PFAM. On posa "*hmm_name*" ha d'anar el nom que hem obtingut a hmmscan.out.
   hmmfetch /mnt/NFS_UPF/soft/databases/pfam-3/Pfam-A.hmm hmm_name **>** hmmname.hmm

3. **Busca templates** a la base de dades PDB:
   hmmsearch hmmname.hmm /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq **>** name_pdb.out

4. Aconsegueix el **PDB** i la **seqüència** de la template (*1sbh* és un exemple del PDB ID de la possible template, hauriem de buscar la que ens hagués donat a hmmsearch).
   cp /mnt/NFS_UPF/soft/databases/pdb/pdb/data/structures/divided/pdb/sb/pdb*1sbh*.ent.gz **.**
   gunzip pdb2sbh.ent.gz
   PDBtoSplitChain.pl -i pdb1sbh.ent **-o** 1sbh

5. **Alinea** les seqüències de la target i la template:
   cat P11018.fa 1sbhA.fa **>** target_template.fa
   hmmalign hmmname.hmm target_template.fa **>** target_template.sto
   aconvertMod2.pl -in h **-out** p **<**target_template.sto**>**target_template.pir
   modpy.sh python2 modeling.py
   **No oblideu modificar modeling.py si és necessari*

6. Avalua el model amb **PROSA** (igual que hem fet abans)

7. Comprova el model amb **Chimera** (igual que hem fet abans)
   chimera hmm_model.pdb 1sbhA.pdb

FOLD PREDICTION - THREADING (pràct 5.3)

Farem servir el programa THREADER, que utilitza un sistema molt similar a **PSIPRED** i **alinea l'estructura secundària** predita amb un conjunt d'estructures conegudes. Al final avalua els models utilitzant potencials estadístics i dóna uns scores.

THREADER triga com unes 4h. Triga tant perquè el model que fa no es basa en homologia sinó que es basa en threading. En el **threading** es força que una seqüència adopti un folding concret d'una estructura disponible a PDB. Es creen molts models i, per tant, després s'han d'anar **avaluant** per escollir el millor. En aquesta pràctica compararem el model que nosaltres hem predit amb el model que fa THREADER (que ens donen ja fet).

THREADER

1. Crear una llista amb els codis dels arxius que hi ha a $THREAD_DIR (un directori amb coses pel THREADER). Aquí fem servir la mateixa target que al modelling, que sabem que es una **serine-proteinasa.** Per tant, a la base de dades de Threader seleccionarem cadenes que també siguin **proteinases.**

   grep PROTEINASE $THREAD_DIR/tdb/* **| fgrep -v** INHIBITOR **| cut –d " " -f 2 >** mylist.txt.

   Així, estem **buscant** els els **noms** de les seqüències de les proteïnes que continguin la paraula proteïnasa, i que NO continguin la paraula inhibidor, i ho estem **ficant** a mylist.txt.

2. Per fer la **comparació** de l'estructura secundària, primer hem d'obtenir l'estructura secundària amb **PSIPRED**.

   psipred P11018.fa

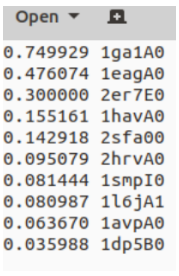
3. Fer **THREADER** amb la predicció de PSIPRED i la llista dels noms de les seqüències amb que compararem.

   threader -pm -v -j P11018.horiz results.ssout mylist.txt **>** results.sslog

   Aquí hem fet servir les opcions **-v, -j i -pm** per obtenir tots els tests possibles (**-j**), extreure tota la *verbose information* **(-v**) i guardar tots els alineaments de Threader en format Modeller (**-pm**).
   Després d'aquesta comanda obtenim dos outputs:
   - results.ssout: conté els scores. El progama mai sobreescriurà aquest document.
   - results.sslog: conté l'alineament en format pir, preparat per fer Modeller.

4. Utilitzar l'expert mode per analitzar i fer un ranking dels millors scores, així no els has de comparar tots.

   $THREADER_DIR/texp/texp **-s** $THREADER_DIR/texp/weights.dat results.ssout **| sort –n –r >** results.texp



5. També podem fer Threader sense utilitzar l'estructura secundària prèvia amb la comanda:

   threader –pm –v –j target.fa results.out mylist.lst **>&** threader.log

   A threader.log trobem diferents columnes amb informació: a la columna 6 hi ha les energies de les interaccions entre residus, a la columna 7 hi ha els z-scores…

6. Ara s'ha de generar un **model** de target.fa utilitzant els millors candidats de results.out. Com a punt de partida s'ha d'utilitzar l'alineament de Threader (threader.log). Aquest ja està en format pir, així que no l'hem de convertir, Modeller ja el reconeix. Per fer servir modeller necessitarem importar al nostre directori el python amb les comandes:

   cp /mnt/NFS_UPF/soft/modeller/examples/automodel/model-default.py **.**

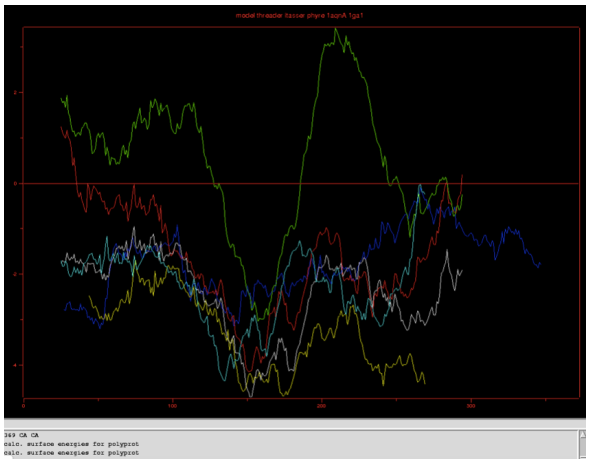
7. Construcció de nous models amb Modeller (= que pràct 4.3)

PHYRE, iTASSER, FUGUE i MODLINK

PHYRE, iTASSER, FUGUE i MODLINK són programes molt similars a THREADER. El guó de pràctiques explica una mica com fer-los servir, però a la pràctica (com que triguen molt a fer-se) ja hi havien models fets a partir d'aquests programes. El que fem es **comparar** els nostre model fet amb Modeller i el model fet amb Phyre, iTasser, Fugue i ModLInk.

Un cop més, per comparar fem servir **PROSA**. Hem de crear una sessió (sessionT.cmd) que contingui el següent:

```
Open
read pdb hmodel1.pdb model
read pdb threader.B99990001.pdb threader
read pdb i-tasser_model1.pdb itasser
read pdb phyre_intensive.pdb phyre
read pdb 1aqnA.pdb 1aqnA
read pdb 1ga1_A.pdb 1ga1
analyse energy *
winsize * 50
shift 1aqnA +19
shift 1ga1 +2
color * model cyan
color * threader green
color * itasser white
color * phyre red
color * 1ga1 blue
init zscore
zscore * z-result
plot
```



D'aquesta manera obtenim els perfils energètics i els z-scores, i podem comparar quin model surt millor.

PRÀCT 6: DOCKING - HEX i FTDOCK

Tenim dues proteïnes i sabem que aquestes interaccionen, pero no sabem com és aquesta interacció o quines regions hi estàn implicades. Per trobar com és la conformació d'aquestes proteïnes quan interaccionen es fan servir programes de **docking**. A més a més, també es pot reconstruir la interacció entre dos proteïnes **superimposant** les estructures de proteïnes homòlogues que també interaccionin.

Els **programes de docking** exploren l'espai conformacional de les interaccions entre dues proteïnes. Després, obtenen una sèrie de possibles conformacions d'aquella interacció i les posen en un ranking segons el seu score. Aquestes diferents conformacions també s'anomenen ***docking poses***. Hi ha diferents algoritmes per obtenir les *docking poses*. Es divideixen en dos grans grups: els que es basen en **propietats fisico-químiques** i els **knowledge-based** (com els potencials estadístics).

Principalment, hi ha dos tipus d'algoritmes per a docking:

- **Rigid body docking:** obté les *docking poses* movent i rotant les estructures com si fossin rígides. Això significa que no s'introdueixen canvis estructurals en les molècules.

- **Flexible docking:** funciona més o menys com l'anterior però permet cert grau de flexibilitat en les molècules, de manera que s'introdueixen alguns canvis conformacionals.

També hem de tenir en compte que els estudis de docking es poden aplicar a interaccions entre altres tipus de molècules, com per exemple interaccions entre una proteïna i el DNA.

En estudis de docking, sempre hi ha una molècula que anomenem **receptor** i una altra que anomenem **lligand**. Normalment el receptor és més gran que el lligand.

En aquesta pràctica utilitzarem els programes **HEX** i **FTDOCK** per fer estudis de **docking**,

Docking amb HEX

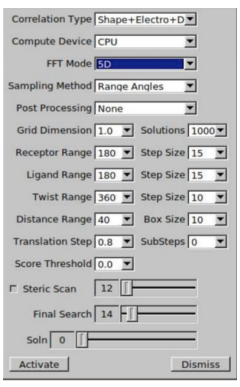
Per obrir i utilitzar hex necessitem aquesta comanda.

hex cpa_receptor.pdb pci_ligand.pdb **-l** docking.log

Aquí hem posat com a input el **receptor** i el **lligand** però, hi ha vegades que també disposem d'un pdb que conté les dues estructures **interaccionant**. Aquest també el podem incloure a la comanda de la següent manera:

hex cpa_receptor.pdb pci_ligand.pdb 4cpa.pdb **–l** docking_4cpa.log

Si volem fer servir HEX d'una manera més avançada, podem canviar els **paràmetres del docking**. La manera més eficient de fer això és utilitzar els **macros**. Els **macros** són arxius que contenen diferents comandes que HEX necessita per funcionar amb els paràmetres que nosaltres vulguem. La manera més fàcil de generar un macro és executar HEX, introduir els paràmetres que ens interessin i guardar-los com un macro. Per fer-ho comencem escrivint **hex** al terminal. Un cop s'obri hex fem el següent:



1. Clica a ***macros***. Després a ***learn***. Això activa la opció learn de HEX.
   *Aquest pas és **molt important:** quan HEX està en learn vol dir que està "aprenent" tot allò que facis a continuació i al final ho podrà guardar. Si t'oblides de posar learn, després quan guardis el macro aquest estarà buit, perquè HEX no haurà après res.*

2. Obre les estructures del **lligand** i el **receptor**. Se t'ha d'obrir una finestra en la qual pots modificar els paràmetres de HEX. Has de posar els

paràmetres igual que es mostra a la imatge de la dreta.

3. Selecciona el rang d'estructures que vulguis guardar. Clica a ***file*** i a ***save***, després a ***range***. Això vol dir que si seleccionem un range de 100, el programa guardara les 100 millors *doscking poses* segons els seus scores.

4. Guarda el summary ***file***. Clica a file i a ***save***, després a ***summary***. El summary conté els scores i les energies de les docking poses que HEX ha generat.

5. Guarda el macro. Clica a ***macros*** i a ***save***. Com que teniem activada la opció de learn, els canvis que hem fet i els paràmetres s'han guardat en un arxiu macro.

6. Ara podem executar directament el macro que acabem de guardar des del terminal:

   hex <macro_ file**>** macro.log

Docking amb FTDOCK

FTDOCK és una mica més difícil d'utilitzar que HEX però és més versàtil.

1. Els PDB han de ser **pre-processats** abans de fer-los servir com a input a FTDOCK. Ho fem amb un programa perl:

   preprocess-pdb.perl -pdb cpa_receptor.pdb
   preprocess-pdb.perl -pdb pci_ligand.pdb

2. Ara, hem de **canviar el nom** de les cadenes d'aquests PDBs a L pel lligand i R pel receptor:

   change-pdb-chain-id.perl -pdb cpa_receptor.parsed **-old ' ' -new 'R'**
   change-pdb-chain-id.perl -pdb pci_ligand.parsed **-old 'I' -new 'L'**

3. Ara ja podem **executar** FTDOCK:

   ftdock –static cpa_receptor.parsed **–mobile** pci_ligand.parsed **> output &**

4. El problema és que FTDOCK triga més d'un dia a acabar la feina. Per això hem de matar el programa i utilitzar uns arxius que ja ens donen. Com matar el programa:

   ps → ens diu quins programes s'estan executant. Hem de mirar el job ID del FTDOCK.
   kill -9 job_ID
   ftdock -rescue → ens rescata els outputs precomputats. Els ouputs són scratch_parameters.dat i scratch_scores.dat.

5. La última comanda triga com 5 minuts a realitzar-se i ens donarà un arxiu que es diu ftdock_global.dat. Aquest conté un **ranking** dels resultats de docking segons els scores. Per analitzar aquest resultats fem servir el programa **rpscore**. Rpscore donarà uns altres scores per diferenciar les docking poses correctes i les incorrectes:

   rpscore -in ftdock_global.dat **-out** ftdock_rpscored.dat **-matrix** best.matrix

6. A cada execució de FTDOCK, s'estan creant moltes *docking poses*. Una manera de **filtrar-les** és seleccionar aquelles en què certs aminoàcids de les dos molècules estan a una certa distància. Això és útil perquè en molts casos tenim alguna **informació** (trobada a la literatura) sobre quins aminoàcids estan implicats en la interacció. Aquí tenim un exemple on filtrem les *docking poses* en què els residus 38 del lligand i 120 del receptor estan a 8 Angstroms:

   filter –constraints L38:R120 **–distance** 8.0 **–out** filter1.dat

7. Podem **generar estructures** per les *docking poses* que hem filtrat. Aquí en generem 10:

   build –in filter1.dat **–b2** 10

   **També es pot fer amb les estructures sense filtrar:*
   build –in *ftdock_rpscored.dat* ***–b2*** *10*

Avaluació dels resultats del Docking

Si tenim l'estructura de la **interacció** entre el receptor i el lligand, podem comprovar com de bons són els resultats del docking. L'avaluació es fa **superimposant** les estructures. Per cada *docking pose*, s'ha de **superimposar la *docking pose*** amb la **interacció** obtinguda **experimentalment**, utilitzant el **receptor** com la **cadena de referència.** Això vol dir que, després de la superimposició, els **receptors** de la docking pose i de l'estructura de referència estan a situats a les **mateixes coordenades**. Llavors, hem de mirar si els **lligands** de les dues interaccions estan a posicions **similars**. Si el **docking és correcte**, els dos lligands estaran molt a prop a la superimposició i tindràn valors de **RMSD** molt **baixos**. Si el **docking no és correcte**, els lligands estaran lluny l'un de l'altre i per tant el seu **RMSD** serà molt **alt**.

Aquesta avaluació es pot fer tant amb FTDOCK com amb HEX, però és molt més fàcil fer-ho amb **Chimera**, sobretot després de fer tot el treball de Bioestructural.

Per fer-ho amb **Chimera**, només has d'obrir el teu docking i el docking experimental i superposar-los amb el **Match Maker**. Has d'agafar la cadena del receptor experimental com a cadena de referència. A més a més, amb Chimera també és fàcil veure el RMSD i per tant avaluar el docking.

Utilitzar la superimposició per reconstruir interaccions domini-domini

Tenim dues proteïnes humanes que en teoria interaccionen: una de la família **Ras** (4fmd_F) i una de la família **RhoGAP** (3byi_A). Per altra banda, tenim la estructura de la **interacció** d'une**s altres proteïnes** de les mateixes famílies i que són **homòlogues** a les humanes (Ras#RhoGAP.4.brk). Tenint això, podem reconstruir la interacció de les proteïnes humanes. Un cop més, al guió de pràctiques posa una manera de fer-ho, però jo recomano fer-ho amb **Chimera**.

El que volem aconseguir és un PDB de la interacció entre les dues proteïnes humanes. Llavors podem obrir a Chimera el PDB de la interacció (Ras#RhoGAP.4.brk) i els dos PDBs del receptor i el lligand (4fmd_F i 3byi_A). Amb ***MatchMaker*** ho superimposem tot, llavors quedarà com és la interacció entre les humanes a sobre la interacció experimental (perquè si són homòlogues coincidiran bastant). Al final, com que volem el PDB de la interacció humana només, anem a ***Tools → General Controls → Command Line.*** Quan surti la barra del *Command Line* escrivim **delete #model0** (en el cas que la interacció experimental sigui el model0). Ara ja podem guardar el PDB de la interacció humana.

Si ens pregunten si això podria passar a la natura, hem de mirar que cap dels àtoms del lligand i el receptor es trobin a les mateixes coordenades exactes. Si no se superposen, podria passar a la natura. Si ho fan, segurament no passa.