Bioinformática Estructural

Amaranta Manrique de Lara y Ramirez, Valeria Erendira Mateo Estrada

4 de marzo de 2016

Tarea 3: Estructura terciaria

En este módulo se habló sobre la relación entre la secuencia de una proteína y su estrucutra terciaria. También se trató de observar el efecto de esto sobre los alineamientos cuando se busca comparar un par de proteínas; cuando la estructura está más conservada que la secuencia, es posible realizar búsquedas de relaciones filogenéticas.

En esta práctica se dispone de la secuencia de una proteína A y se quiere conocer - de manera aproximada - su estructura tridimiensional.

Para solucionar este problema se propone la siguiente solución: Estimar coordenadas cartesianas para la mayoría de los átomos de A, en base a la estructura conocida de proteínas similares, llamadas moldes, tempaldos o *templates*.

3.1) Seleccionar una superfamilia de proteínas de SCOP y extraer la secuencia de aminoácidos (ATOM) y las coordenadas PDB de varios dominios de la misma.

Seleccionamos una acuaporina y cuatro proteínas homólogas provenientes de diferentes especies.

Identificador	Organismo de procedencia
d1h6ia	Humano
1rc2	Oveja
d1ymga1	Vaca
d1j4na	Vaca
1sor	*Escherichia coli*

La acuaporina *query*, o de interés, es la proteína **d1h6ia** de humano. La comparación se realizó con los cuatro archivos PDB de las estructuras de las proteínas homólogas. La secuencia query fue obtenida de la SCOPE y las secuencias de las posibles proteínas homólogas fueran obtenidas de **SCOPE** y de **Protein Data Bank**.

3.2) Comprobar que las secuencias descargadas coincidan con las coordenadas.

Para comprobar que las secuencias coincidan con las coordenadas se utilizó el siguiente programa:

```
# File with 3D coords of insulin
 my $pdb_file = "2ins.pdb";

 # Extract coords of the alpha carbons of the chain A, and alpha and beta carbons of histidines of chai
 my @desired_coords = ('a/*/ca/', 'b/*/ca|cb/his');

 open(PDBFILE,$pdb_file);
 my $pdb_content = join('',<PDBFILE>);
 close PDBFILE;

 my $pdb_coords = join('',extract_pdb_coords($pdb_content,\@desired_coords));
```

```
open(PDBFILE,">$pdb_file.out");
print PDBFILE $pdb_coords;
close PDBFILE;

# Extract desired PDB coordinates from PDB entries
sub extract_pdb_coords {

     my ($pdb_content, $types) = @_;

     my $pdb_data;
     my $patterns;
     my @pattern_parts = ('chain','res_number','res_type','res_name');
     foreach my $type (@{$types}) {
          my $count = 0;
          my %pattern;
          while ($type =~ /([^\/]+)/g){
               $type = $'; # Text to the right of the match
               $pattern{$pattern_parts[$count]} = $1;
               $count++;
          }
          push(@{$patterns},\%pattern);
     }

     foreach my $pattern (@{$patterns}){
          my ($chain,$res_number,$res_type,$res_name);
          if (!defined($pattern->{'chain'}) || !$pattern->{'chain'} || $pattern->{'chain'} eq '*'){
               $chain = '\w{1}';
          } else {
               $chain = uc($pattern->{'chain'});
          }
          $pattern->{'res_number'} =~ s/\s+//;
          if (!defined($pattern->{'res_number'}) || !$pattern->{'res_number'} || $pattern->{'res_numbe
               $res_number = '\d+';
          } elsif ($pattern->{'res_number'} =~ /^(\d+)$/) {
               $res_number = $1;
          } elsif ($pattern->{'res_number'} =~ /^(\d+)-(\d+)$/) {
               $res_number = '('.join('|', $1 .. $2).')';
          } elsif ($pattern->{'res_number'} =~ /\d,/) {
               $res_number = '('.join('|', split(",",$pattern->{'res_number'})).')';
          }
          if (!defined($pattern->{'res_type'}) || !$pattern->{'res_type'} || $pattern->{'res_type'} eq
               $res_type = '[\w\d]+';
          } else {
               $res_type = uc($pattern->{'res_type'});
          }
          if (!defined($pattern->{'res_name'}) || !$pattern->{'res_name'} || $pattern->{'res_name'} eq
               $res_name = '\w{3}';
          } else {
               $res_name = uc($pattern->{'res_name'});
          }
          $pattern = '/(ATOM|HETATM)\s+\d+\s+('.$res_type.')\s+('.$res_name.')\s('.$chain.')\s+('.$res
     }

     my @pdb_data_lines = extract_lines_from_text($pdb_content, $patterns);
```

```
     if (@pdb_data_lines){
          $pdb_data = join("\n",@pdb_data_lines);
     }

     return $pdb_data;

}

# Extract lines from text with the desired patterns
sub extract_lines_from_text {

     my ($text, $patterns) = @_;

     my @data;
     my @lines = split("\n",$text);

     foreach my $line (@lines){
          foreach my $pattern (@{$patterns}){
               if ($pattern =~ /^\/(.+)\/$/){
                    if ($line =~ /$1/){
                         push(@data,$line);
                         last;
                    }
               } else {
                    if ($line =~ /\Q$pattern\E/){
                         push(@data,$line);
                         last;
                    }
               }
          }
     }
```

3.3) Calcular al menos dos alineamientos pareados entre secuencias de aminoácidos extraídas y calcular su porcentaje de identidad (total de parejas de residuos idénticas / total parejas alineadas).

Se obtuvo la identidad utilizando pBLAST para comparar cada una de las acuaporinas y la secuencia de la proteína query. Los resultados se mostrarán en el siguiente punto.

3.4) Calcular con MAMMOTH los alineamientos estructurales de los dominios alineados en base a su secuencia. Visualizar con Rasmol.

Se comparó la proteína con cada una de las secuencias utilizando **MAMMOTH**, empleando los siguientes comandos:

```
mammoth -p d1h6ia.pdb -e 1rc2.pdb -o log.out
mammoth -p d1h6ia.pdb -e d1ymga1.pdb -o log.out
mammoth -p d1h6ia.pdb -e 1j4na.pdb -o log.out
mammoth -p d1h6ia.pdb -e 1sor.pdb -o log.out
```

De esto se obtuvieron tres archivos de salida:

- maxsub_sup.pdb

- maxsub_sup2.pdb
- rasmol.tcl

A cada uno de los archivos resultantes se le añadió un tag para distiguir los organismos de procedencia.

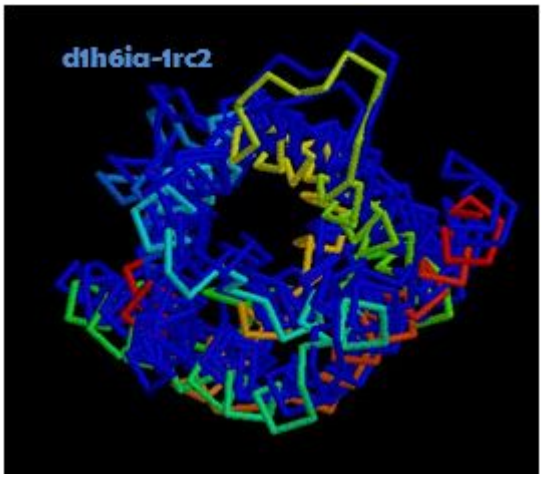
Terminación (tag)	Organismos de procedencia
ho	Humano-oveja
hv	Humano-vaca (d1ymga1)
hv2	Humano-vaca (d1j4na)
hc	Humano-*E. coli*

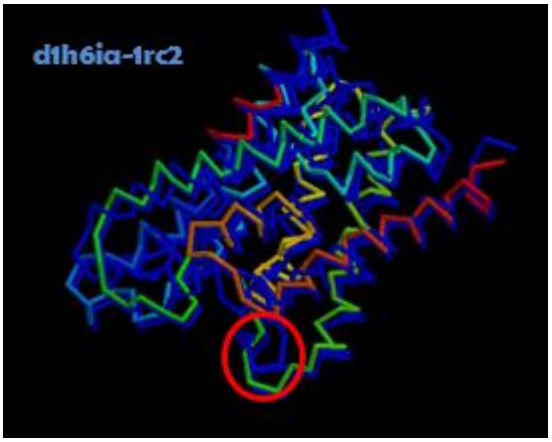
Los E-values y z-scores fueron obtenidos de cada una de las comparaciones utilizando MAMMOTH. La identidad (3.4), E-value y z-score están resumidos en la tabla que se presenta a continuación:

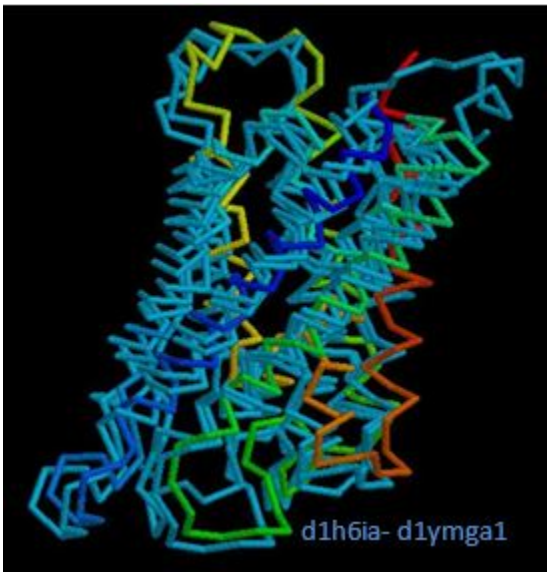
Archivo	E-value	Z-score	Identidad
ho	0.20072832E-11	28.317269	36%
hv	0.40097925E-11	27.578202	49%
hv2	0.87496677E-12	29.204150	90%
hc	0.30403458E-11	27.873827	49%

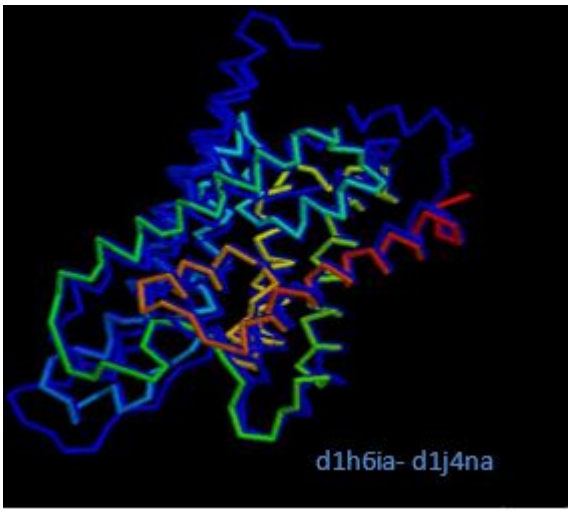
3.5) Comparar los alineamientos obtenidos en 3.3 y 3.4. Comentar en qué elementos de estructura secundaria se observan diferencias.

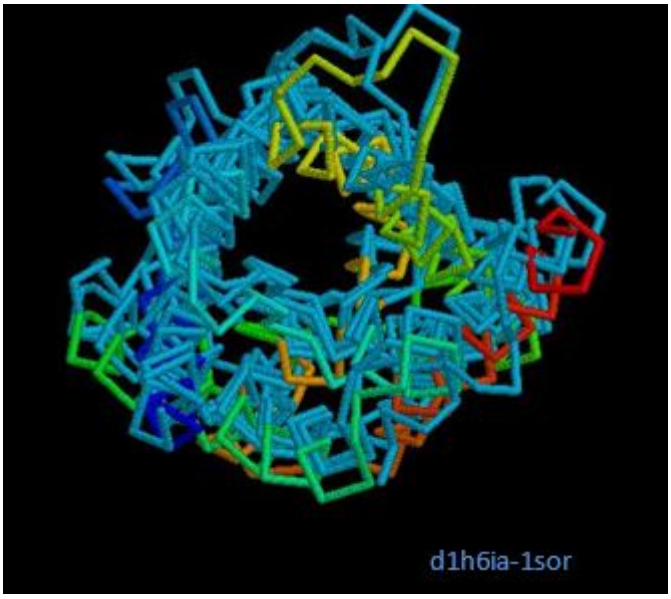
Posteriormente se visualizaron las estructuras usando **Rasmol**; se sobrepuso la secuencia query con cada una de las otras proteínas. Se puede observar que son muy similares ya que sobrelapan estructuralmente, sin embargo difieren en algunas regiones. Algunos loops son asimétricos. La proteína query es la que se muestra en diferentes colores (cada dominio esta e un color diferente) y la proteína contra la que se está comparando se muestra en azul.




d1h6ia-1rc2


d1h6ia- d1ymga1


d1h6ia- d1j4na



3.6) Utilizar el prog3.1 para calcular el error (RMDSD) de los alineamientos.

A continuación se presentan los resultados de correr el programa *prog3.1*.

d1h6ia-1sor

total residuos: pdb1 = 225 pdb2 = 235

total residuos alineados = 218

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

RMSD = 4.90 Angstrom

d1h6ia-1rc2

total residuos: pdb1 = 225 pdb2 = 231

total residuos alineados = 217

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

RMSD = 3.44 Angstrom

d1h6ia-dj1ymga1

total residuos: pdb1 = 225 pdb2 = 233

total residuos alineados = 218

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

RMSD = 3.30 Angstrom

d1h6ia-dj4na

total residuos: pdb1 = 225 pdb2 = 249

total residuos alineados = 224

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

RMSD = 13.58 Angstrom

¿Son mejores o peores los alineamientos basados en secuencia desde el punto de vista del RMSD?

De acuerdo al z-score la acuaporina d1j4na es la más parecida a la secuencia problema. Este resultado también es el mejor resultado de acuerdo al RMSD, ya que tiene el número más bajo y por lo tanto difiere menos de la proteína query: se adapta mejor a la estructura.
