Tarea III: Algoritmos 3D-3

Citlali Gil Aguillon.

Jessica Danielly Medina Sánchez.

Anali Migueles Lozano.

1) Selecciona una superfamilia de proteínas de SCOP (http://scop.berkeley.edu) y extrae la secuencia de aminoácidos (ATOM records) y las coordenadas PDB de varios dominios de la misma. Podéis ver un ejemplo de dominio en http://scop.berkeley.edu/sunid=29763, y abajo están tanto la secuencia como una liga para descargar las coordenadas.

Class: Alpha proteins

Super family: Histone-fold

Family: Nucleosome core histones

Species: Human (Homo sapiens), H2A.a

- SCOPe Domain Sequences for d2cv5a_:
- SCOPe Domain Sequences for d2cv5c1_:
- SCOPe Domain Sequences for d2cv5d1_:

Species: Mouse (Mus musculus), H2A.1

- SCOPe Domain Sequences for d2f8na_:
- SCOPe Domain Sequences for d2f8ne_:

2) Comprueba que las secuencias descargadas coinciden con las coordenadas.

Se utilizo el siguiente código para corroborar que las secuencias fueran las mismas.

```
#!/usr/bin/perl
#  Extract sequence chains from PDB file
use strict;
use warnings;
# Read in PDB file:  Warning - some files are very large!
my @file = get_file_data('pdb/c1/pdb1c1f.ent');

# Parse the record types of the PDB file
my %recordtypes = parsePDBrecordtypes(@file);
```

```
# Extract the amino acid sequences of all chains in the protein
my @chains = extractSEQRES( $recordtypes{'SEQRES'} );

# Translate the 3-character codes to 1-character codes, and print
foreach my $chain (@chains) {
    print "****chain $chain **** \n";
    print "$chain\n";
    print iub3to1($chain), "\n";
}
exit;
#####Subrotinas#####
# parsePDBrecordtypes
#--given an array of a PDB file, return a hash with
#    keys   = record type names
#    values = scalar containing lines for that record type

sub parsePDBrecordtypes {

    my @file = @_;
    use strict;
    use warnings;

    my %recordtypes = (  );

    foreach my $line (@file) {

        # Get the record type name which begins at the
        # start of the line and ends at the first space
        my($recordtype) = ($line =~ /^(\S+)/);

        # .= fails if a key is undefined, so we have to
        # test for definition and use either .= or = depending
        if(defined $recordtypes{$recordtype} ) {
            $recordtypes{$recordtype} .= $line;
        }else{
            $recordtypes{$recordtype} = $line;
        }
    }
```

```
    return %recordtypes;
}

# extractSEQRES
#
#--given an scalar containing SEQRES lines,
#    return an array containing the chains of the sequence

sub extractSEQRES {

    use strict;
    use warnings;

    my($seqres) = @_;

    my $lastchain = '';
    my $sequence = '';
    my @results = (  );

    # make array of lines
    my @record = split ( /\n/, $seqres);

    foreach my $line (@record) {
        # Chain is in column 12, residues start in column 20
        my ($thischain) = substr($line, 11, 1);
        my($residues)  = substr($line, 19, 52); # add space at end

        # Check if a new chain, or continuation of previous chain
        if("$lastchain" eq "") {
            $sequence = $residues;
        }elsif("$thischain" eq "$lastchain") {
            $sequence .= $residues;

        # Finish gathering previous chain (unless first record)
        }elsif ( $sequence ) {
            push(@results, $sequence);
            $sequence = $residues;
        }
        $lastchain = $thischain;
```

```
    }

    # save last chain
    push(@results, $sequence);

    return @results;
}

# iub3to1
#
#--change string of 3-character IUB amino acid codes (whitespace separated)
#    into a string of 1-character amino acid codes

sub iub3to1 {

    my($input) = @_;

    my %three2one = (
      'ALA' => 'A',
      'VAL' => 'V',
      'LEU' => 'L',
      'ILE' => 'I',
      'PRO' => 'P',
      'TRP' => 'W',
      'PHE' => 'F',
      'MET' => 'M',
      'GLY' => 'G',
      'SER' => 'S',
      'THR' => 'T',
      'TYR' => 'Y',
      'CYS' => 'C',
      'ASN' => 'N',
      'GLN' => 'Q',
      'LYS' => 'K',
      'ARG' => 'R',
      'HIS' => 'H',
      'ASP' => 'D',
      'GLU' => 'E',
    );
```

```
    # clean up the input
    $input =~ s/\n/ /g;

    my $seq = '';

    # This use of split separates on any contiguous whitespace
    my @code3 = split(' ', $input);

    foreach my $code (@code3) {
        # A little error checking
        if(not defined $three2one{$code}) {
            print "Code $code not defined\n";
            next;
        }
        $seq .= $three2one{$code};
    }
    return $seq;
}
```

3) Calcula al menos dos alineamiento pareados entre secuencias de aminoácidos de las extraídas en 1 y calcula su %identidad como el total de parejas de residuos idénticas / total parejas alineadas.

Los alineamientos fueron obtenidos utilizando blast

- La identidad entre las dos proteínas alineadas en el primer alineamiento : Humano (d2cv5a) y Ratón (d2f8na) fue de 99%.

d2f8na_ a.22.1.1 (A:) Histone H3 {African clawed frog (Xenopus laevis) [TaxId: 8355]}
Sequence ID: lcl|Query_123769 Length: 98 Number of Matches: 1

Range 1: 1 to 97 Graphics ▼ Next Match ▲ Previous Match

Score	Expect	Method	Identities	Positives	Gaps
195 bits(496)	2e-70	Compositional matrix adjust.	96/97(99%)	96/97(98%)	0/97(0%)

```
Query  1   PHRYRPGTVALREIRRYQKSTELLIRKLPFQRLVREIAQDFKTDLRFQSSAVMALQEACE  60
           PHRYRPGTVALREIRRYQKSTELLIRKLPFQRLVREIAQDFKTDLRFQSSAVMALQEA E
Sbjct  1   PHRYRPGTVALREIRRYQKSTELLIRKLPFQRLVREIAQDFKTDLRFQSSAVMALQEASE  60

Query  61  AYLVGLFEDTNLCAIHAKRVTIMPKDIQLARRIRGER  97
           AYLVGLFEDTNLCAIHAKRVTIMPKDIQLARRIRGER
Sbjct  61  AYLVGLFEDTNLCAIHAKRVTIMPKDIQLARRIRGER  97
```

• La identidad entre las dos proteínas alineadas en el primer alineamiento : Humano (d2cv5a) y Ratón (d2f8na) fue de 28%

d2f8ne_ a.22.1.1 (E:) Histone H3 {African clawed frog (Xenopus laevis) [TaxId: 8355]}
Sequence ID: lcl|Query_254383 Length: 98 Number of Matches: 2

Range 1: 46 to 92 Graphics ▼ Next Match ▲ Previous Match

Score	Expect	Method	Identities	Positives	Gaps
18.1 bits(35)	0.015	Compositional matrix adjust.	13/47(28%)	20/47(42%)	0/47(0%)

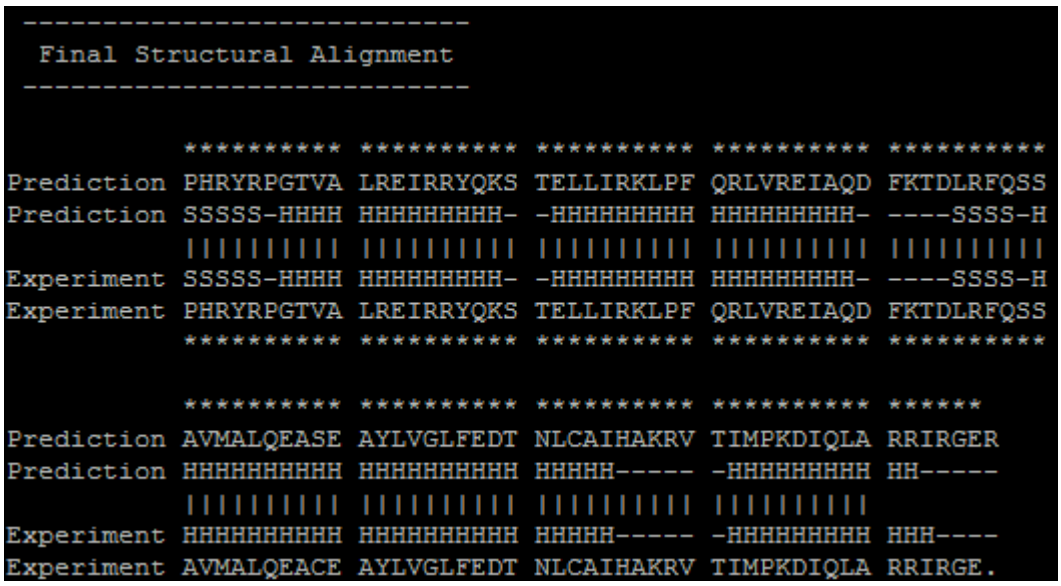
```
Query  32  RVGAGAPVYLAAVLEYLTAEILELAGNAARDNKKTRIIPRHLQLAIR  78
           R  + A + L    E     + E     A   K+  I+P+ +QLA R
Sbjct  46  RFQSSAVMALQEASEAYLVGLFEDTNLCAIHAKRVTIMPKDIQLARR  92
```

Range 2: 81 to 85 Graphics ▼ Next Match ▲ Previous Match ▲ First Match

Score	Expect	Method	Identities	Positives	Gaps
11.2 bits(17)	4.6	Compositional matrix adjust.	2/5(40%)	4/5(80%)	0/5(0%)

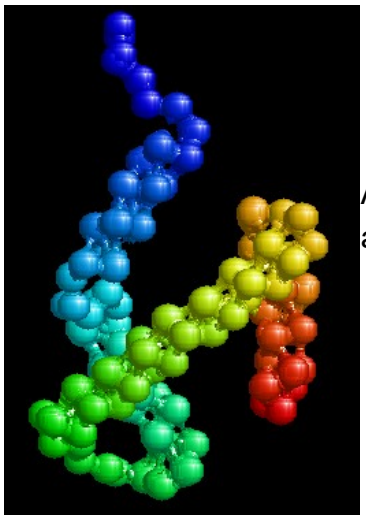
```
Query  104  VLLPK  108
            ++PK
Sbjct  81   TIMPK  85
```

4) Calcula con mammoth los alineamientos estructurales de los dominios que ya alineaste en 3 en base a su secuencia. Visualízalos con Rasmol como se explica en http://eead-csic-compbio.github.io/bioinformatica_estructural/node32.html. El software está en /home/compu2/algoritmos3D/soft/mammoth-1.0-src para que lo copien y compilen con gfortram como se explica en README, cambiando g77 por gfortran.



Primer alineamiento.

Humano (d2cv5a) y Ratón (d2f8na) con identidad de 99%.



Aquí podemos visualizar el archivo que se genero tras el alineamiento. (rasmol.tcl)

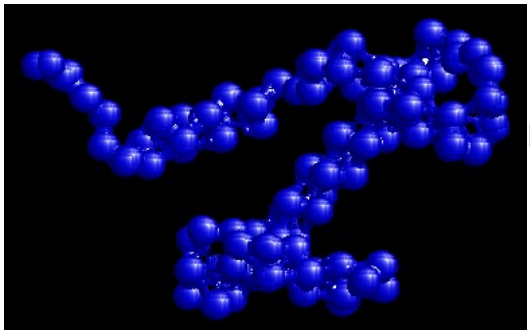


Imagen del archivo MaxSub_sup obtenida.

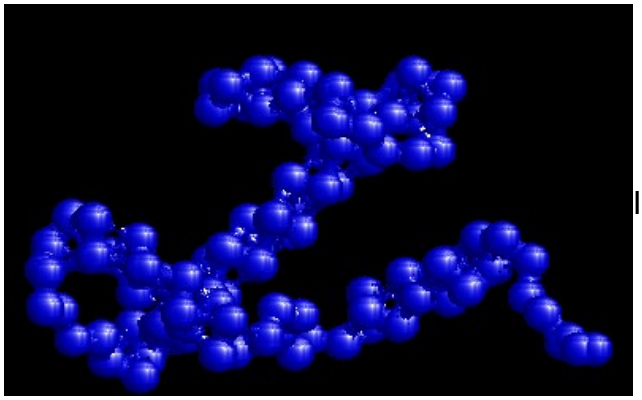


Imagen del archivo MaxSub_sup2 obtenida.

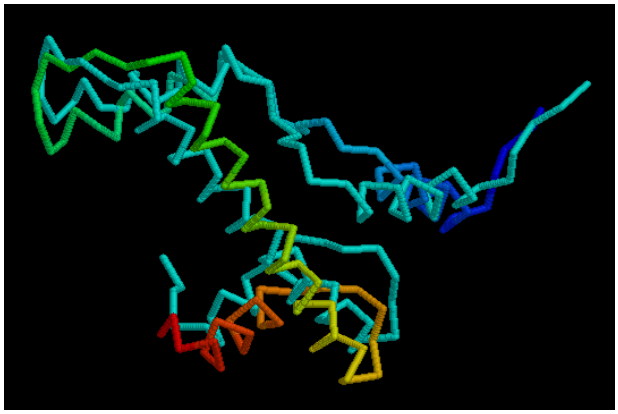
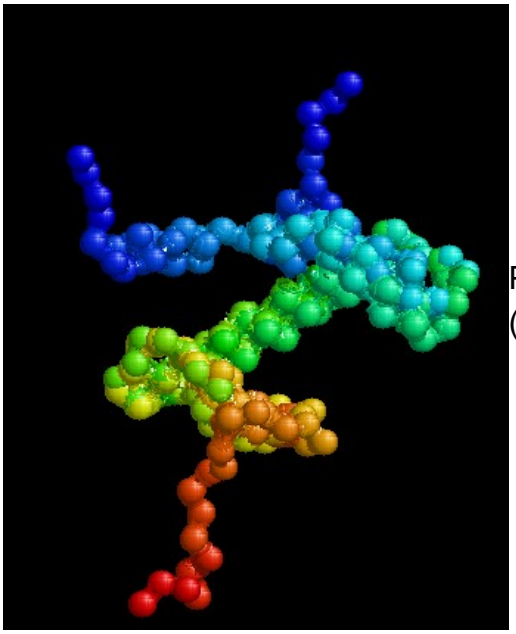


Imagen del alineamiento d2cv5a VS d2f8na

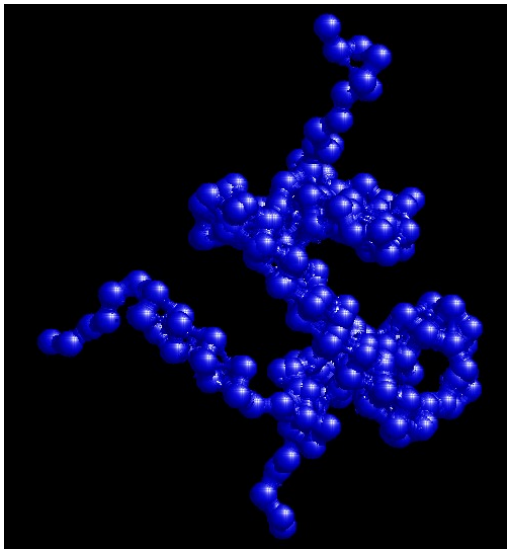
Segundo alineamiento.

Humano (d2cv5c1) y Ratón (d2f8na) fcon identidad de 28%

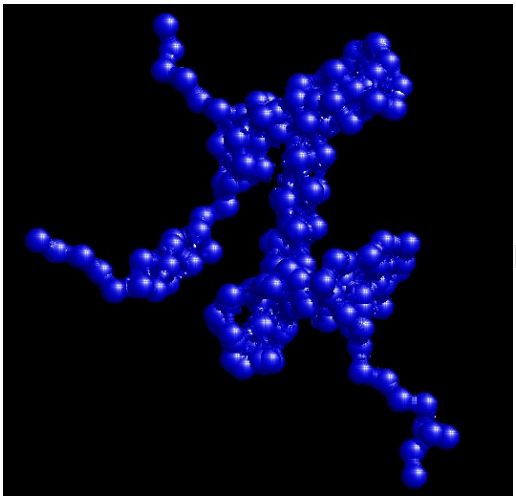
```
----------------------------
 Final Structural Alignment
----------------------------

                                  ********* *    ****  **********
Prediction .........R AKAKTRSS.R AGLQFPVGRV H....RLLRK GNYSERVGAG
Prediction ---------S SSSSSSSS-- ----HHHHHH H----HHHHH -----SSS-H
                        |||| | |||||||||| |    ||      ||||||||
Experiment SSSSS-HHHH HHHHHHHHH- -HHHHHHHHH HHHHHHHHH- ----SSSS-H
Experiment PHRYRPGTVA LREIRRYQKS TELLIRKLPF QRLVREIAQD FKTDLRFQSS
                                  ********* *    ****  **********

           ********** ********** ********** ********** ******
Prediction APVYLAAVLE YLTAEILELA GNAARDNKKT RIIPRHLQLA IRNDEELNKL
Prediction HHHHHHHHHH HHHHHHHHHH HHHHH----- -HHHHHHHHH HHHH-HHHHH
           |||||||||| |||||||||| |||||||||| |||||||||| |
Experiment HHHHHHHHHH HHHHHHHHHH HHHHH----- -HHHHHHHHH HH--SSS---
Experiment AVMALQEASE AYLVGLFEDT NLCAIHAKRV TIMPKDIQLA RRIRGER...
           ********** ********** ********** ********** ******
```



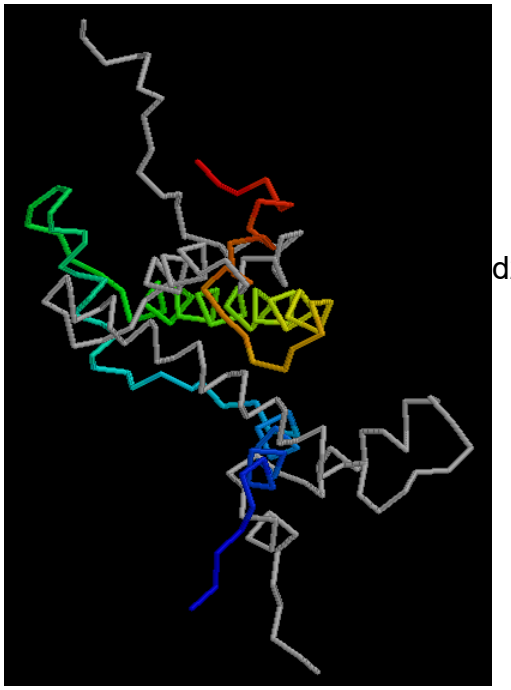
Podemos visualizar la Imagen obtenida tras el alineamiento. (rasmol.tcl)



MaxSub_sup



MaxSub_sup2



d2cv5a VS d2f8ne

5) Compara los alineamientos obtenidos en 3 y 4. Comenta en qué elementos de estructura secundaria se observan diferencias.

Se observan diferencias en las hélices alfa y las laminas beta en las estructuras secundarias de las proteínas. Aunque en el primer alineamiento aparecen menos también se pueden visualizar.

6) Utiliza el prog3.1 (en http://eead-csic-compbio.github.io/bioinformatica_estructural/node31.html) para calcular el error (RMSD) de los alineamientos obtenidos en 3 y 4 y comenta los resultados. Son mejores o peores los alineamientos basados en secuencia desde el punto de vista del RMSD?

Los alineamientos de secuencia son menos significativos, los alineamientos basados en estructura nos dan mejores resultados ya que se acercan más a la realidad porque contemplan las cargas entre los enlaces, repulsiones e interacciones entre sus grupos funcionales y cadenas.

Resultados del alineamiento 1

total residuos: pdb1 = 97 pdb2 = 98

total residuos alineados = 96

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

 # RMSD = 0.33 Angstrom

Resultados del alineamiento 2

total residuos: pdb1 = 108 pdb2 = 98

total residuos alineados = 83

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

 # RMSD = 7.91 Angstrom

Prueba MAMMOTH con el fin de comparar una estructura problema (una de las 5) contra una biblioteca de estructuras en formato PDB (las otras 4), como si fuera BLAST. Cuál de las 4 estructuras sería el mejor molde o *template*) Cuál es el límite esperado de precisión, en términos de RMSD, que alcanzaríamos con cada molde?

Alineamiento de d2cv5a (estructura problema) con

1. d2f8na
2. d2cv5c1
3. d2cv5d1
4. d2f8ne

Alineamiento MAMMOTH **d2cv5a vs d2f8na**

total residuos: pdb1 = 97 pdb2 = 98

total residuos alineados = 96

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

 # RMSD = 0.33 Angstrom

Alineamiento MAMMOTH **d2cv5a vs d2cv5c1**

total residuos: pdb1 = 97 pdb2 = 108

total residuos alineados = 71

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

 # RMSD = 12.80 Angstrom

Alineamiento MAMMOTH **d2cv5a vs d2cv5d1**

total residuos: pdb1 = 97 pdb2 = 96

total residuos alineados = 74

coordenadas originales = original.pdb

superposicion optima:

archivo PDB = align_fit.pdb

 # RMSD = 4.33 Angstrom

Alineamiento MAMMOTH **d2cv5a vs d2f8ne**

total residuos: pdb1 = 97 pdb2 = 98
total residuos alineados = 96
coordenadas originales = original.pdb
superposicion optima:
archivo PDB = align_fit.pdb
 # RMSD = 0.39 Angstrom

El valor de RMSD nos indica la divergencia entre las proteínas por lo tanto el alineamiento con menor valor de RMSD con nuestra proteína será el mejor template. En este caso la proteína con el menor valor de RMSD es: d2f8na la cual presenta un valor de RMSD de 0.33 Angstrom y esta proteína también presenta una identidad del 99%. Por lo tanto por ambos métodos podemos ver que esta proteína es el mejor template para nuestra proteína problema.