Tarea algoritmos3D-3

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 enhttp://scop.berkeley.edu/sunid=29763 , y abajo están tanto la secuencia como una liga para descargar las coordenadas.

http://scop.berkeley.edu/sunid=95078

http://scop.berkeley.edu/sunid=30598

http://scop.berkeley.edu/sunid=30604

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

Corrimos un script en Perl y efectivamente las secuencias coincidían

```
# File with 3D coords of insulin
 my $pdb_file = "d1ps9a3.pdb";

 # Extract coords of the alpha carbons of the chain A, and alpha and beta carbons of histidines of chain B
 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_number'} eq '*'){
            $res_number = '\d+';
        } elsif ($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_number.')\s+.+/';
    }

    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;
                }
            }
        }
    }

    return @data;

}
```

Original Script

1	N	THR A	6	-0.684	14.408	-13.144	1.00	44.27	N	ATOM	2	CA	THR A	6	0.645	14.236	-12.482	1.00	41.63	C
2	CA	THR A	6	0.645	14.236	-12.482	1.00	41.63	C	ATOM	9	CA	PRO A	7	0.212	12.593	-9.110	1.00	31.89	C
3	C	THR A	6	0.498	14.137	-10.968	1.00	38.12	C	ATOM	16	CA	GLN A	8	2.351	14.224	-6.386	1.00	23.10	C
4	O	THR A	6	0.488	15.117	-10.219	1.00	37.72	O	ATOM	25	CA	ILE A	9	3.544	11.906	-3.641	1.00	20.91	C
5	CB	THR A	6	1.682	15.288	-12.884	1.00	43.77	C	ATOM	33	CA	CYS A	10	5.771	12.647	-0.705	1.00	20.75	C
6	OG1	THR A	6	2.838	15.187	-12.026	1.00	45.14	O	ATOM	39	CA	VAL A	11	7.805	10.258	1.448	1.00	18.44	C
7	CG2	THR A	6	1.092	16.679	-12.814	1.00	45.79	C	ATOM	46	CA	VAL A	12	9.058	11.173	4.954	1.00	17.81	C
8	N	PRO A	7	0.344	12.904	-10.520	1.00	34.72	N	ATOM	53	CA	GLY A	13	12.392	9.424	5.638	1.00	15.27	C
9	CA	PRO A	7	0.212	12.593	-9.110	1.00	31.89	C	ATOM	57	CA	SER A	14	15.057	8.186	3.188	1.00	14.82	C
10	C	PRO A	7	1.482	13.049	-8.368	1.00	28.05	C	ATOM	63	CA	GLY A	15	15.865	4.811	4.793	1.00	14.55	C
11	O	PRO A	7	2.593	12.828	-8.838	1.00	25.62	O	ATOM	67	CA	PRO A	16	14.932	1.511	3.084	1.00	14.35	C
12	CB	PRO A	7	0.047	11.087	-9.078	1.00	32.90	C	ATOM	74	CA	ALA A	17	11.207	2.085	3.722	1.00	17.18	C
13	CG	PRO A	7	-0.298	10.663	-10.468	1.00	33.21	C	ATOM	79	CA	GLY A	18	11.173	5.499	1.998	1.00	15.83	C
14	CD	PRO A	7	0.335	11.676	-11.369	1.00	35.13	C	ATOM	83	CA	PHE A	19	13.249	4.336	-0.948	1.00	16.03	C
15	N	GLN A	8	1.288	13.719	-7.240	1.00	26.60	N	ATOM	94	CA	TYR A	20	11.429	1.037	-1.546	1.00	17.46	C
16	CA	GLN A	8	2.351	14.224	-6.386	1.00	23.10	C	ATOM	106	CA	THR A	21	8.079	2.841	-1.287	1.00	18.82	C

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.

Hicimos los alineamientos con MUSCLE

```
#  Percent Identity  Matrix - created by Clustal2.1

     1: d1cjca2     100.00   21.35
     2: d1ps9a3      21.35  100.00


     1: d1cjca2     100.00   22.28
     2: d1djna3      22.28  100.00
```

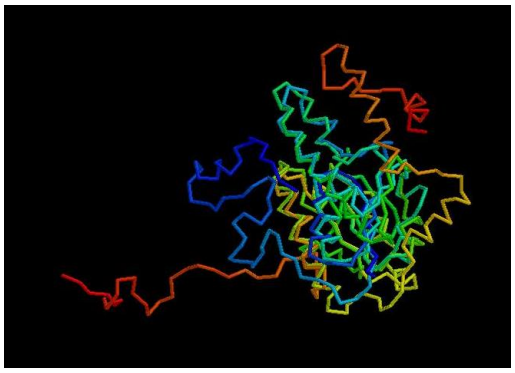
Y el total de parejas alineadas salió con Mammoth

	Residuos Totales	Alineadas
D1cjca2	230	124
D1ps9a3	179	

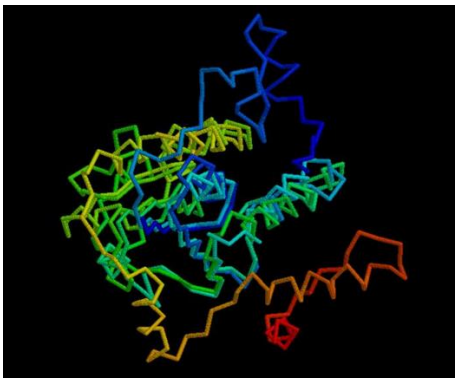
	Residuos Totales	Alineadas
D1cjca2	230	121
D1djna3	233	

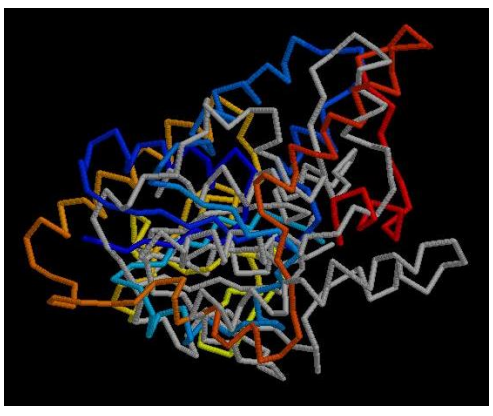
4,5 y 6) white→d1cjca2

d1djna3



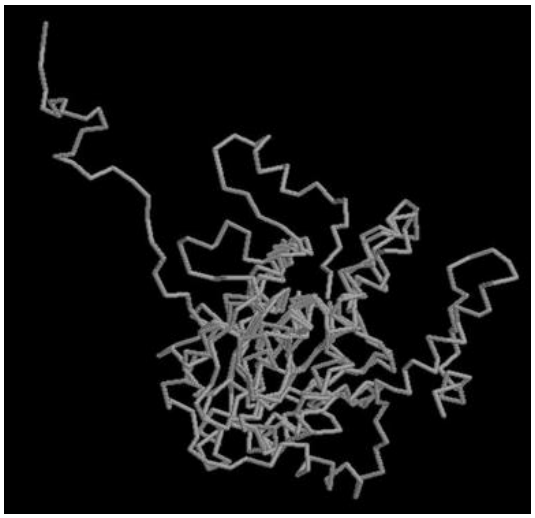
d1ps9a3

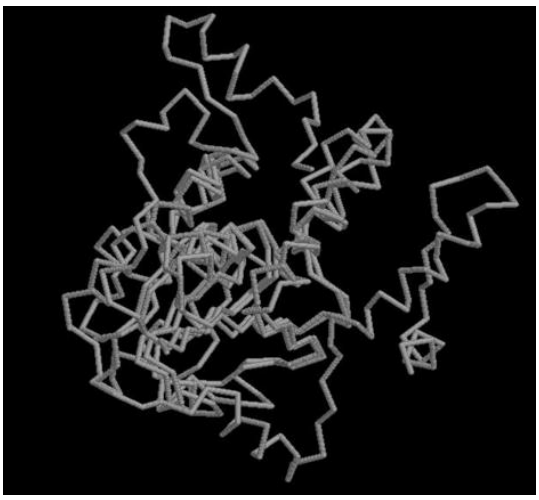




MaxSub1

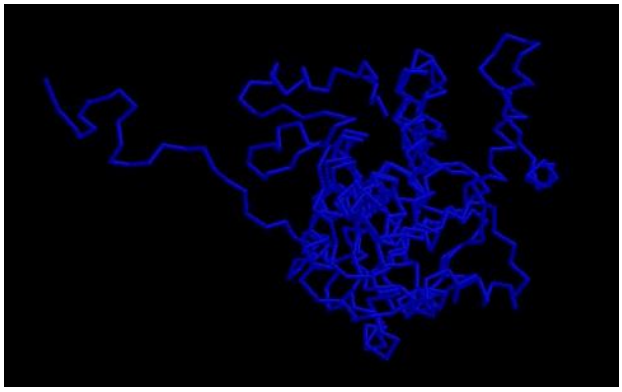
d1djna3 **d1ps9a3**

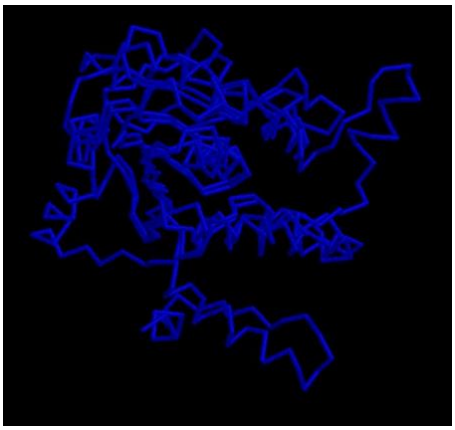




MaxSub2

d1djna3 **d1p s9a3**





D1cjca2 vs D1ps9a3 RMSD = 10.24

D1cjca2 vs D1djna3 RMSD = 8.80

Podemos observar que con d1djna3 intersecta mucho mejor que con d1ps9a3 y también su RMSD es mejor aunque difieren en varias regiones.

Dado que los valores obtenidos a partir de Clustal no toman en cuenta varias propiedades tridimensionales de polipéptidos que MAMMOTH sí, me parece que es una mejor opción usar MAMMOTH ya que muchas proteínas suelen plegarse de maneras similares y parece ser una mejor opción para alineamientos estructurales.