

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
use Math::Trig ;
use strict;
```

```
#{my %coor,my $chnum}=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"O"}}{ my $qqa=substr($r,0,1); if ( $qqa ne $ch){ $chnum++; $ch=$qqa; }
my %qqa=find_quart( %coor{"O"});
```

```
if ($chnum)
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename=~ s/^\./\./;
$filename=~ s/\./\./;
#$filename=$chnum.".".$qnum.".".$filename.".dat";
$filename="$dir".".$filename.".dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $qnum\n";
```

```
foreach my $m (sort { $a<=>$b } keys %coor){
my %qartets;
my %q= find_q( %coor{$m} );
```

```
# foreach my $q { keys %qartets } { print join " ",@{ $qartets{$q} },"\n";
foreach my $q { keys %qartets}{
```

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

```
foreach my $res (@{ $qartets{$q} }){
# print "$q coor{$m}{ $res }{"N"}->x,"\n";
```

```
$nx=$nx+ $coor{$m}{ $res }{"N9"}->x;
$ny=$ny+ $coor{$m}{ $res }{"N9"}->y;
$nz=$nz+ $coor{$m}{ $res }{"N9"}->z;

$ox=$ox+ $coor{$m}{ $res }{"O6"}->x;
$oy=$oy+ $coor{$m}{ $res }{"O6"}->y;
$oz=$oz+ $coor{$m}{ $res }{"O6"}->z;
$r=$res;
}
```

# Структурная Биоинформатика

## Лекция 10. Базы данных: SCOP, CATH; соревнование CASP

Головин А.В.<sup>1</sup>

<sup>1</sup>МГУ им М.В. Ломоносова, Факультет Биотехнологии и Биотехнологии

Москва, 2012

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Содержание

```
#!/(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
```

## Домены

```
foreach my $chnum
keys %{$coor{"0"}}{ my $ggg=substr($r,0,1); if ( $ggg ne $ch){$chnum++; $ch=$ggg } };
my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;
```

## Топология

```
my $filename=$ARGV[0];
$filename=~ s/^\.//;
$filename=~ s/\.pdb//;
my $filename="$chnum"."$qnum"."$filename.dat";
```

## Архитектура

```
print "filename=$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $qnum\n";
```

## SCOP

```
foreach my $m (sort { $a<=>$b } keys %coor){
my $qts = %qwa; #find_quart( %coor{$m} );
my %q=find_q( %coor{$m} );
```

## CATH

```
foreach my $q ( keys %qartets){
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

## CASP

```
foreach my $res (@{ $qartets{$q} }){
print "$q $coor{$m}{$res}{\"N\"}->x,\"n\";
$nx=$nx+ $coor{$m} {$res}{\"N9\"}->x;
$ny=$ny+ $coor{$m} {$res}{\"N9\"}->y;
$nz=$nz+ $coor{$m} {$res}{\"N9\"}->z;

$ox=$ox+ $coor{$m} {$res}{\"O6\"}->x;
$oy=$oy+ $coor{$m} {$res}{\"O6\"}->y;
$oz=$oz+ $coor{$m} {$res}{\"O6\"}->z;
```



```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Структурный домен (биоинформатика)

```
#!/(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } ;
```

```
my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;
```

```
if ($qnum > 0){
```

```
my $qnum="mkdir $ARGV[1]";
```

```
my $qnum="mkdir $ARGV[1]";
```

```
$filename=$ch.$chnum;
```

```
$filename=$ch.$chnum;
```

```
$filename=$dir.$filename.".dat";
```

```
print "filename=$filename";
```

```
open OUT, ">$filename";
```

```
print OUT "INFO Chain $chnum $qnum $qnum\n";
```

```
foreach my $m ( keys %coor ){
```

```
my %qwa=find_quart( %coor{"$m"} );
```

```
my %q= find_q( %coor{"$m"} );
```

```
# foreach my $q ( keys %qartets ){ print join " ", @{$qartets{$q}} ,"\n";
```

```
foreach my $q ( keys %qartets ){
```

```
my $nx; my $ny; my $nz;
```

```
my $ox; my $oy; my $oz;
```

```
my $r;
```

```
foreach my $res ( @{$qartets{$q}} ){
```

```
# print "$q $coor{$m}{$res}{\"N\"}->x,\"n\";
```

```
$nx=$nx+ $coor{$m}{$res}{\"N9\"}->x;
```

```
$ny=$ny+ $coor{$m}{$res}{\"N9\"}->y;
```

```
$nz=$nz+ $coor{$m}{$res}{\"N9\"}->z;
```

```
$ox=$ox+ $coor{$m}{$res}{\"O6\"}->x;
```

```
$oy=$oy+ $coor{$m}{$res}{\"O6\"}->y;
```

```
$oz=$oz+ $coor{$m}{$res}{\"O6\"}->z;
```

Обособленная в пространстве часть белка, его структурная единица, имеющая:

- сравнительно мало контактов с другими частями белка
- собственное гидрофобное ядро

#!/usr/bin/perl

use Math::VectorReal qw( // );

# Домен белка XXX (жизнь)

#(my %\$coor,\$my \$snum)=read\_pdb(\$ARGV[0]);

my %\$coor=read\_pdb(\$ARGV[0]);

my \$dir=\$ARGV[1];

my \$ch,\$my \$snum;

foreach my \$r ( sort keys %{\$coor{"0"}} ){ my \$ggg=substr(\$r,0,1); if ( \$ggg ne \$ch ){ \$snum++; \$ch=\$ggg } };

my %\$qwa=find\_quart( %\$coor{"0"} ); my %\$snum=keys %\$qwa;

## Часть белка, названная доменом:

if (\$snum &gt; 0){

my \$filename=\$ARGV[0];

\$filename=~ s/^.\*\.//;

\$filename=~ s/\./\_//;

#\$filename=\$snum."\_".\$snum."\_".\$filename.".dat";

\$filename=\$snum."\_".\$snum."\_".\$filename.".dat";

print "file: \$filename";

open OUT,"&gt;\$filename";

print OUT "#INFC chain \$snum qnum \$snum\n";

foreach my \$m ( sort { \$a &lt;=&gt; \$b } keys %\$coor ){

my %\$quartets = %\$qwa; #find quart( %\$coor{\$m} );

my %\$q = %\$quartets;

# foreach my \$q ( keys %\$quartets ){ print join " ", @{\$quartets{\$q}}, "\n";

foreach my \$q ( keys %\$quartets ){

my \$nx; my \$ny; my \$nz;

my \$ox; my \$oy; my \$oz;

my \$r;

foreach my \$res ( @{\$quartets{\$q}} ){

# print "\$q \$coor{\$m}{\$res}{\"N\"-&gt;x,\"n\";

\$nx=\$nx+ \$coor{\$m}{\$res}{\"N9\"-&gt;x};

\$ny=\$ny+ \$coor{\$m}{\$res}{\"N9\"-&gt;y};

\$nz=\$nz+ \$coor{\$m}{\$res}{\"N9\"-&gt;z};

\$ox=\$ox+ \$coor{\$m}{\$res}{\"O6\"-&gt;x};

\$oy=\$oy+ \$coor{\$m}{\$res}{\"O6\"-&gt;y};

\$oz=\$oz+ \$coor{\$m}{\$res}{\"O6\"-&gt;z};

• Субъективизм

• Образность

• Традиция

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Пример

```
#!/my $coor,my $chnum)=read_pdb($ARGV[0]);
```

**В полимеразах обычно выделяют три домена: fingers, palm, thumb**

```
my $ch, my $chnum;
sort keys %{$coor{"0"}}( my $gggg=substr($r,0,1); if ( $gggg ne $ch){ $chnum++; $ch=$gggg } );
my %qwa=find_quart( $coor{"0"} ); my $sqnum=keys %qwa;
```

```
if ($sqnum > 0){
```

```
#system("mkdir
```

```
my $filename=;
```

```
$filename="-- s/
```

```
$filename="-- s/
```

```
#$filename=$c
```

```
$filename="$di
```

```
print "$filenam
```

```
open OUT,">$fi
```

```
print OUT "#INF
```

```
foreach my $m
```

```
my %qartets
```

```
my %q= find
```

```
# foreach m
```

```
foreach m
```

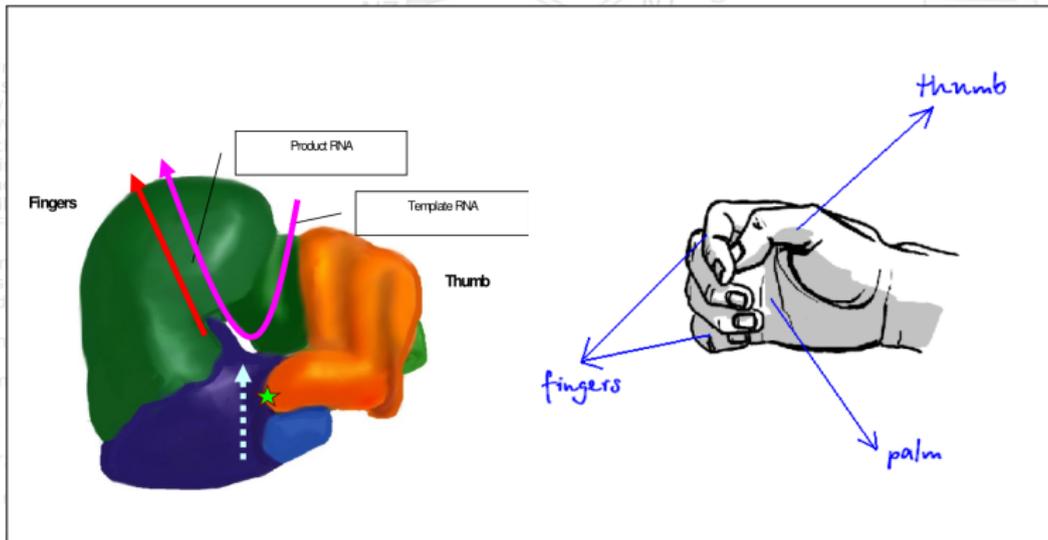
```
my $nx;
```

```
my $ox;
```

```
my $r
```

```
foreach
```

```
#
```



```
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

## Итог

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){ $chnum++; $ch=$ggg } };

```

```
my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;
```

```

if ($qnum >0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename=~ s/^.*\///;
$filename=~ s/\.pdb//;
#
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $qnum\n";

```

```

foreach my $m (sort {$a<=>$b} keys %coor){
my %qartets = %qwa; #find_quart( %coor{$m} );
my %q= find_q( %coor{$m} );

```

```
# foreach my $q ( keys %qartets){ print join " ",@{$qartets{$q}},"\n";
```

```
foreach my $q ( keys %qartets){
```

```

my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;

```

```
foreach my $res ( @{$qartets{$q}}){
```

```
#
print "$q $coor{$m}{$res}{"$r"}->x,\n";
$nx=$nx+ $coor{$m}{$res}{"N9"}->x;
$ny=$ny+ $coor{$m}{$res}{"N9"}->y;
$nz=$nz+ $coor{$m}{$res}{"N9"}->z;

```

```

$ox=$ox+ $coor{$m}{$res}{"O6"}->x;
$oy=$oy+ $coor{$m}{$res}{"O6"}->y;
$oz=$oz+ $coor{$m}{$res}{"O6"}->z;

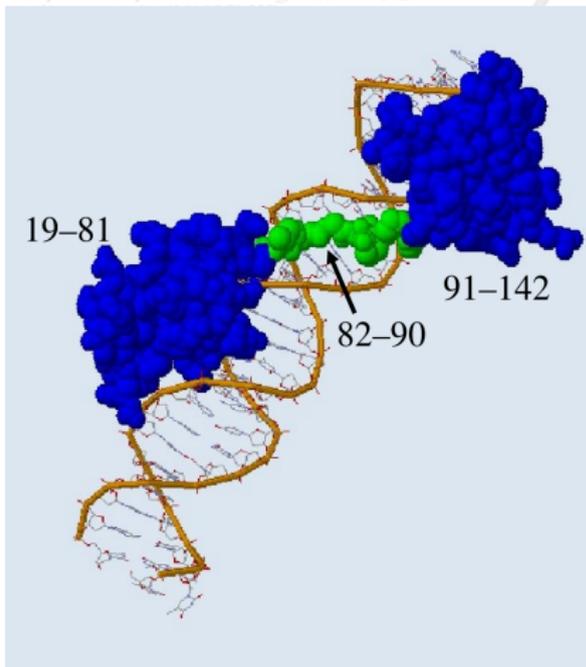
```

Три определения доменов часто дают похожие результаты!  
Но не всегда

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Пример

```
$(my %coor,$my $schnum)=read_pdb($ARGV[0]);
```



«Парный» (“Paired”) домен из транскрипционного фактора PAX5 человека (PDB 1K78) – очевидно, два структурных домена

Эволюционный домен (PAX в Pfam) включает оба структурных домена (126 а.о.)

```
$nx=$nx+ $coor{$m}{$res}{"N9"}->x;
$ny=$ny+ $coor{$m}{$res}{"N9"}->y;
$nz=$nz+ $coor{$m}{$res}{"N9"}->z;
```

```
$ox=$ox+ $coor{$m}{$res}{"O6"}->x;
$oy=$oy+ $coor{$m}{$res}{"O6"}->y;
$oz=$oz+ $coor{$m}{$res}{"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Пример

```

#(my %$coor,my $schnum)=read_pdb($ARGV[0]);
my %$coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $sch ){ $schnum++; $sch=$ggg } ;

```

```
my %$qwa=find_quart( $coor{"0"} ); my $sqnum=keys %$qwa;
```

**Забавно, что полипептидные цепи обоих структурных доменов имеют общую топологию**

- одинаковое число спиралей,
- одинаковые межспиральные взаимодействия,
- одинаковый порядок следования спиралей вдоль цепи;
- \* минорные элементы вторичной структуры не в счет!

```

foreach my $q ( keys %$qartets ){
    my $nx; my $ny; my $nz;
    my $ox; my $oy; my $oz;
    my $r;

    foreach my $res ( @{$ $qartets{$q} } ){
        #
        print "$q $coor{$$m}{$$res}{"$R"}->x,"$n";
        $nx=$nx+ $coor{$$m}{$$res}{"N9"}->x;
        $ny=$ny+ $coor{$$m}{$$res}{"N9"}->y;
        $nz=$nz+ $coor{$$m}{$$res}{"N9"}->z;

        $ox=$ox+ $coor{$$m}{$$res}{"O6"}->x;
        $oy=$oy+ $coor{$$m}{$$res}{"O6"}->y;
        $oz=$oz+ $coor{$$m}{$$res}{"O6"}->z;
    }
}

```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Структурные домены, Алгоритмы

```

#(my %$coor,my $schnum)=read_pdb($ARGV[0]);
my %$coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $sch ){ $schnum++; $sch=$ggg } ;

```

```
my %$qwa=find_quart( %$coor{"0"} ); my $qnum=keys %$qwa;
```

## Основы методов

- Домен имеет собственное гидрофобное ядро (пример: алгоритм DETECTIVE Swindells, 1995)
- Домен – это часть белка, внутри которой много контактов аминокислотных остатков, а между доменами – мало контактов (пример: алгоритм DOMAK, Siddiqui&Barton, 1995)

```

foreach my $q ( keys %$qartets ){
    my $nx; my $ny; my $nz;
    my $ox; my $oy; my $oz;
    my $r;

    foreach my $res ( @{$ $qartets{$q} } ){
        print "$q $coor{$m}{$res} {"$r"}->x,"$n";
        $nx=$nx+ $coor{$m}{$res} {"$r"}->x;
        $ny=$ny+ $coor{$m}{$res} {"$r"}->y;
        $nz=$nz+ $coor{$m}{$res} {"$r"}->z;

        $ox=$ox+ $coor{$m}{$res} {"O6"}->x;
        $oy=$oy+ $coor{$m}{$res} {"O6"}->y;
        $oz=$oz+ $coor{$m}{$res} {"O6"}->z;
    }
}

```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Siddiqui & Barton, 1995: DOMAK

```
#!/my $coor,my $chnum)=read_pdb($ARGV[0]);
my $coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } };
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
```

Сверху – вниз, от целого – к части!

- Предпосылки: домен состоит из одного или двух непрерывных участков полипептидной цепи
- Число контактов между остатками внутри домена больше, чем число междоменных контактов

```
# foreach my $q ( keys %qartets ){ print join " ",@{$qartets{$q}} ,"\n";
```

```
foreach my $q ( keys %qartets ){
```

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

```
foreach my $res ( @{$qartets{$q}} ){
```

```
# print "$q $coor{$m}{$res} {"N"}->x,"n";
```

```
$nx=$nx+ $coor{$m}{$res} {"N9"}->x;
```

```
$ny=$ny+ $coor{$m}{$res} {"N9"}->y;
```

```
$nz=$nz+ $coor{$m}{$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m}{$res} {"O6"}->x;
```

```
$oy=$oy+ $coor{$m}{$res} {"O6"}->y;
```

```
$oz=$oz+ $coor{$m}{$res} {"O6"}->z;
```





# Алгоритм

- К полной цепи применяются 2 метода (целый или сегментный домен). Выбирается разделение с лучшим индексом
- К полученным двум доменам применяется та же процедура. В случае, когда домен состоит из двух сегментов, применяется также дополнительный метод.
- Алгоритм останавливается в зависимости от пороговых значений:
  - MDS – минимальный размер домена (в числе остатков)
  - MSS – минимальный размер сегмента
- Отдельная процедура предусмотрена для сегментов, длина которых между MDS и MSS
- Найденные домены проверяются на “компактность”; некомпактные – сливаются в один

# Swindells, 1995 DETECTIVE

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

my ($my $coor, my $schnum)=read_pdb($ARGV[0]);
my $my $coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $sggg=substr($r,0,1); if ( $sggg ne $sch ){ $schnum++; $sch=$sggg } ;
```

```
my $my $qwa=find_quart( $coor{"0"} ); my $qnum=keys %$qwa;
if ($qnum > 0){
  #system("mkdir $ARGV[1]");
  my $filename=$ARGV[0];
  $filename="-- s/ /_/g";
  $filename="$mdir/$filename.dat";
  print "$filename\n";
  open OUT,">$filename";
  print OUT "#INFO chain $schnum qnum $qnum\n";
  foreach my $m (sort { $a<=>$b } keys %coor){
    my $my $qartets = %qwa; #find quart( $coor{$m} );
    my $my $q
```

**Снизу – вверх, наращивание частей!**

**Предпосылка: каждый домен имеет свое гидрофобное ядро.**

**Этапы:**

- выявление гидрофобных ядер в структуре
- «натягивание» доменов на гидрофобные ядра

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;

foreach my $res ( @{$qartets{$q}} ){
  # print "$q $coor{$m} {$res} {"N1"}->x,\n";
  $nx=$nx+ $coor{$m} {$res} {"N1"}->x;
  $ny=$ny+ $coor{$m} {$res} {"N1"}->y;
  $nz=$nz+ $coor{$m} {$res} {"N1"}->z;

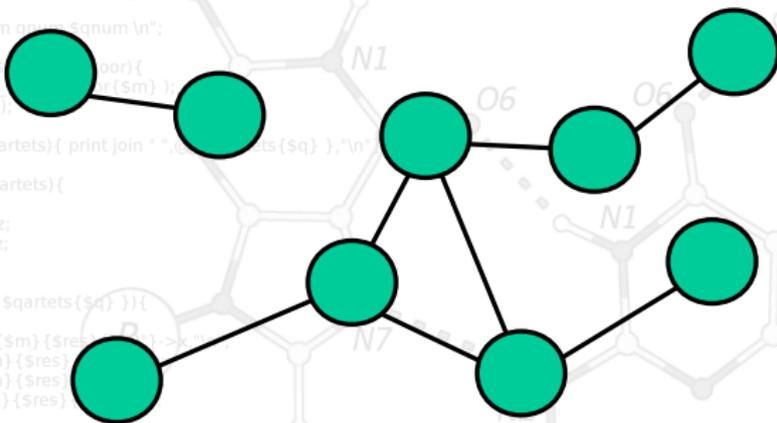
  $ox=$ox+ $coor{$m} {$res} {"O6"}->x;
  $oy=$oy+ $coor{$m} {$res} {"O6"}->y;
  $oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```



# Подробности

Строится граф:

- Вершина – отобранный остаток
- Ребро соединяет вершины, если соответствующие остатки гидрофобно взаимодействуют
- Связные компоненты графа, содержащие 5 или более остатков, называются гидрофобными ядрами



# Всё сложнее

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){$chnum++; $ch=$ggg} };
```

```
my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;
```

```
if ($qnum > 0){
```

## Гидрофобные ядра – еще не домены!

Для получения доменов применяется много ходовая процедура  
ЧИСТКИ-СЛИЯНИЯ

```
# foreach my $q { keys %qartets } { print join " ", @{$qartets{$q}}, "\n";
```

```
foreach my $q { keys %qartets } {
```

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

```
foreach my $res ( @{$qartets{$q}} ) {
```

```
# print "$q $coor{$m}{$res} {"N"}->x, "\n";
```

```
$nx=$nx+ $coor{$m}{$res} {"N9"}->x;
```

```
$ny=$ny+ $coor{$m}{$res} {"N9"}->y;
```

```
$nz=$nz+ $coor{$m}{$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m}{$res} {"O6"}->x;
```

```
$oy=$oy+ $coor{$m}{$res} {"O6"}->y;
```

```
$oz=$oz+ $coor{$m}{$res} {"O6"}->z;
```

# Топология

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
my ($R) = @ARGV;
my ($qwa) = @ARGV;
my ($qnum) = @ARGV;
my ($qartets) = @ARGV;
```

Это описание последовательности элементов вторичной структуры и их взаимного расположения в пространстве.

```
if ($qnum > 0){
#system("mkdir $ARGV[1]*");
#system("cp $ARGV[0] $ARGV[1]*");
}
```

**Белки имеют одинаковую топологию** если **основные** элементы вторичной структуры расположены в последовательности в одном и том же порядке и взаимное расположение этих элементов в пространстве сходно.

```
foreach my $q { keys %qartets }
```

Термины “укладка” (folding) и “топология” (topology) обычно трактуются как синонимы

```
foreach my $res { @{$qartets{$q}} }{
#
print "$q $coor{$$m}{$$res} {"$R"}->x,"$n";
$nx=$nx+ $coor{$$m}{$$res} {"$R"}->x;
$ny=$ny+ $coor{$$m}{$$res} {"$R"}->y;
$nz=$nz+ $coor{$$m}{$$res} {"$R"}->z;

$ox=$ox+ $coor{$$m}{$$res} {"O6"}->x;
$oy=$oy+ $coor{$$m}{$$res} {"O6"}->y;
$oz=$oz+ $coor{$$m}{$$res} {"O6"}->z;
```



# Архитектура

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

my ($ARGV[0]) = @ARGV;
my ($coor, $schnum) = read_pdb($ARGV[0]);
my $dir = $ARGV[1];
my $sch, $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ) { my $ggg = substr($r, 0, 1); if ( $ggg ne $sch ) { $schnum++; $sch=$ggg } }
```

```
my %qwa = find_quart( $coor{"0"} ); my $qnum = keys %qwa;
```

это описание взаимного расположения в пространстве элементов вторичной структуры без учета их последовательности в полипептидной цепи

```
if ($qnum > 0) {
    $filename = "s/" . $qnum;
    $filename = $dir . $filename . ".dat";
    print "$filename\n";
    open OUT, ">$filename";
    print OUT "#INFO chain $schnum qnum $qnum\n";
}
```

Т.о., белки с одинаковой топологией (укладкой) имеют одинаковую архитектуру, по определению.

Обратное не верно!

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;

foreach my $res ( @{$ $partets{$sq}} ) {
    print "$q $coor{$m}{$res} {"$r"}->x,\n";
    $nx = $nx + $coor{$m}{$res} {"$r"}->x;
    $ny = $ny + $coor{$m}{$res} {"$r"}->y;
    $nz = $nz + $coor{$m}{$res} {"$r"}->z;

    $ox = $ox + $coor{$m}{$res} {"O6"}->x;
    $oy = $oy + $coor{$m}{$res} {"O6"}->y;
    $oz = $oz + $coor{$m}{$res} {"O6"}->z;
}
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

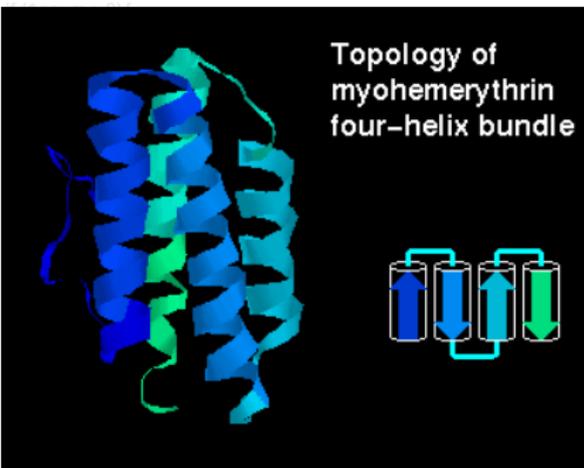
## Пример

```
#!/my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
```

## Архитектура – пучок 4х параллельных спиралей

```
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } }
```

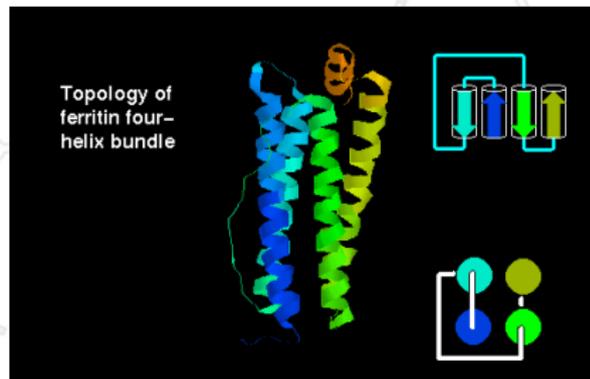
```
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
```



```
# print "$q $coor{$m} {$res} {"$R"}->$x,$n";
$nx=$nx+ $coor{$m} {$res} {"$R"}->$x;
$ny=$ny+ $coor{$m} {$res} {"$R"}->$y;
$nz=$nz+ $coor{$m} {$res} {"$R"}->$z;

$ox=$ox+ $coor{$m} {$res} {"O6"}->$x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->$y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->$z;
```

Топология пучка параллельных спиралей может быть отражена диаграммой TOPS



# Недостаток

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

my ($R) = @ARGV;

my ($coor, $schnum) = read_pdb($ARGV[0]);
my $coor = read_pdb($ARGV[0]);
my $mdir = $ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"O"}} ) { my $gggg = substr($r, 0, 1); if ( $gggg ne $sch ) { $schnum++; $sch = $gggg } }
```

К сожалению, на сегодня отсутствуют адекватные универсальные способы описания топологии ... Поэтому остается большой произвол в трактовке того, какие топологии встречаются в белках, сколько разных топологий и т.п.

В отличие от сравнительной однозначности в трактовке элементов вторичной структуры разными авторами и программами!

```
foreach my $m ( sort { $a <=> $b } keys %coor ) {
    my %quartets = %qwa; #find_quart($coor{$m});
    #foreach my $q ( keys %quartets ) { print join " ", $quartets{$q}, "\n"; }

    my $sx, my $sy, my $sz;
    my $r;

    foreach my $res ( @{$quartets{$q}} ) {
        #
        print "$q $coor{$m}{$res} {"$R"}->x, "\n";
        $nx = $nx + $coor{$m}{$res}{"N9"}->x;
        $ny = $ny + $coor{$m}{$res}{"N9"}->y;
        $nz = $nz + $coor{$m}{$res}{"N9"}->z;

        $ox = $ox + $coor{$m}{$res}{"O6"}->x;
        $oy = $oy + $coor{$m}{$res}{"O6"}->y;
        $oz = $oz + $coor{$m}{$res}{"O6"}->z;
    }
}
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Примеры описания топологии. TOPS

```
#!/(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } };
```

```
my %qwa=find
```

```
if ($qnum > 0) {
#system("mkdir
my $filename=
$filename="-- $
$filename="-- $
# $filename=$
$filename="$c
print "$filename
open OUT,">$
print OUT "#IN
```

```
foreach my $
my %qartets
my %q= find
```

```
# foreach
```

```
foreach my $q ( keys %qartets ) {
```

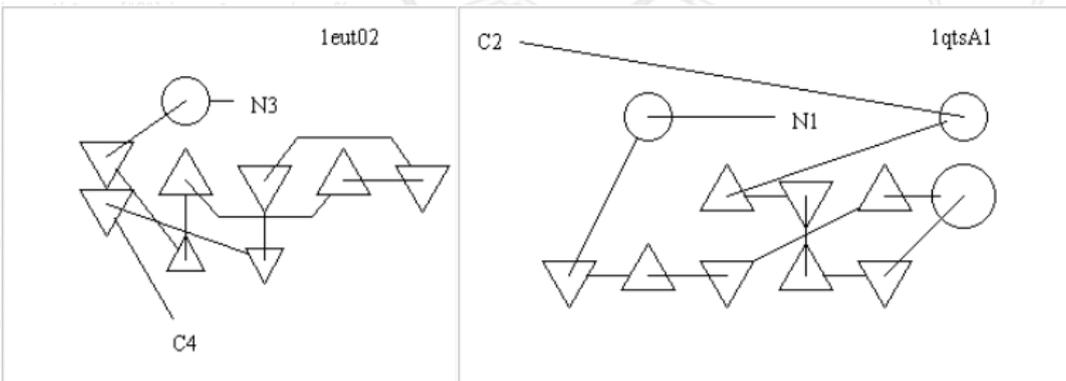
```
my $nx; my $ny; my $nz;
```

## На самом деле, у этих белков сходная топология

```
foreach my $res ( @ { $qartets{$q} } ) {
```

```
# print "$q $coor{$m}{$res} {"N9"}->x,"n";
$nx=$nx+ $coor{$m}{$res} {"N9"}->x;
$ny=$ny+ $coor{$m}{$res} {"N9"}->y;
$nz=$nz+ $coor{$m}{$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m}{$res} {"O6"}->x;
$oy=$oy+ $coor{$m}{$res} {"O6"}->y;
$oz=$oz+ $coor{$m}{$res} {"O6"}->z;
```



```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Примеры, частые в глобулярных белках

```
my ($my $coor, my $chain) = read_pdb($ARGV[0]);
my $coor = read_pdb($ARGV[0]);
my $dir = $ARGV[1];
my $ch = $ARGV[2];
foreach my $chain ($chain) {
```

- Бета-сэндвич (sandwich)

- Бета-баррель (barrel)

- Рулет (jelly roll) – по существу, сэндвич, содержащий мотив “рулет”

- Бета-спираль (beta-helix)

- 3х (4х) спиральный узел (3helical bundle)

- Пучек спиралей (parallel helical bundles)

- Спирализованная спираль

- Бета-цилиндр (TIM barrel)

- Укладка Россмана (Rossmann fold)

```
# print "$q $coor{$m} {$res} {"$N"}->x, "\n";
$nx=$nx+ $coor{$m} {$res} {"$N"}->x;
$ny=$ny+ $coor{$m} {$res} {"$N"}->y;
$nz=$nz+ $coor{$m} {$res} {"$N"}->z;

$ox=$ox+ $coor{$m} {$res} {"$O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"$O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"$O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Классификации структурных доменов

```

#(my %$coor,my $schnum)=read_pdb($ARGV[0]);
my %$coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $sch){ $schnum++; $sch=$ggg } ;

```

```
my %$qwa=find_quart( $coor{"0"} ); my $sqnum=keys %$qwa;
```

```
if ($sqnum > 0){
```

```
  #system("mkdir $ARGV[1]");
```

```
  my $filename;
```

```
  $filename=$ARGV[0].".pdb";
```

```
  #system("cp $ARGV[0].pdb $filename");
```

```
  $filename=$ARGV[0].".pdb";
```

```
  print "$filename\n";
```

```
  open OUT, ">$filename";
```

```
  print OUT;
```

```

foreach my $m (sort { $a<=>$b } keys %$coor){
  my %$qwa=find_quart( $coor{$m} );
  my %$qartets={ keys %$qwa };

```

```
#   foreach my $q ( keys %$qartets){ print join " ",@{$qartets{$q}};,"n";
```

```
   foreach my $q ( keys %$qartets){
```

```
     my $nx; my $ny; my $nz;
```

```
     my $ox; my $oy; my $oz;
```

```
     my $r;
```

```
     foreach my $res ( @{$qartets{$q}} ){
```

```
       #   print "$q $coor{$m} {$res} {"N"}->x,"n";
```

```
       $nx=$nx+ $coor{$m} {$res} {"N9"}->x;
```

```
       $ny=$ny+ $coor{$m} {$res} {"N9"}->y;
```

```
       $nz=$nz+ $coor{$m} {$res} {"N9"}->z;
```

```
       $ox=$ox+ $coor{$m} {$res} {"O6"}->x;
```

```
       $oy=$oy+ $coor{$m} {$res} {"O6"}->y;
```

```
       $oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

- SCOP (Murzin, Benner, Hubbard, Chotia, 1995)
- CATH (Orengo et al., 1993, 1997)
- FSSP (Holm&Sander, 1993)
- другие

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Structural Classification of Proteins, SCOP

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){ $chnum++; $ch=$ggg } };

```

```
my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;
```

```

if ($qnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename="-- s/^.*\//";
$filename="-- s/\ .pdb//";
#file:
$filename="$dir/$filename.dat";
print "$filename\n";
open OUT ">$filename";
print OUT "chain $chnum $qnum $qnum\n";

```

- Экспертное выделение доменов

- Экспертная классификация

```

foreach my $m (sort {$a<=>$b} keys %coor){
my %qartets = %qwa; #find_quart( %coor{$m} );
my %q = find_q( %coor{$m} );

```

```
# foreach my $q ( keys %qartets){ print join " ",@{$qartets{$q}},"\n";
```

```
foreach my $q ( keys %qartets){
```

```

my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;

```

```
foreach my $res ( @{$qartets{$q}}){
```

```
# print "$q $coor{$m}{$res}{\"N\"}->x,\"n\";
```

```
$nx=$nx+ $coor{$m}{$res}{\"N9\"}->x;
```

```
$ny=$ny+ $coor{$m}{$res}{\"N9\"}->y;
```

```
$nz=$nz+ $coor{$m}{$res}{\"N9\"}->z;
```

```
$ox=$ox+ $coor{$m}{$res}{\"O6\"}->x;
```

```
$oy=$oy+ $coor{$m}{$res}{\"O6\"}->y;
```

```
$oz=$oz+ $coor{$m}{$res}{\"O6\"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Уровни классификации в SCOP

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } ;

```

```
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
```

- Класс

```
if ($qnum > 0){
```

- Укладка (fold) – сходная топология

```
#system("cd $dir $ARGV[0]");
```

```
my $filename=$ARGV[1];
```

```
$filename=$dir.$filename;
```

```
$filename=$dir.$filename;
```

```
#system("cd $dir $ARGV[0]");
```

```
$filename=$dir.$filename;
```

```
print "$filename\n";
```

```
open OUT, ">$filename";
```

```
print OUT "class $qnum\n";
```

- Семейство – сходство последовательностей и/или хорошее пространственное выравнивание цепей

```
foreach my $m ( keys %{$coor{"0"}} ){
```

```
my %qartets=find_quart($coor{"0"});
```

```
my %q=keys %qartets;
```

```
# foreach my $q ( keys %qartets ){ print join " ", $qartets{"$q"}, "\n";
```

- Белок – б.м. ортологичные белковые домены

```
foreach my $q ( keys %qartets ){
```

```
my $k=keys %qartets{"$q"};
```

- Вид – конкретный белок

```
my $nx; my $ny; my $nz;
```

```
my $ox; my $oy; my $oz;
```

```
my $r;
```

```
foreach my $res ( @{$qartets{"$q"}} ){
```

```
# print "$q $coor{"$m"} {"$res"} {"$r"}->x, "\n";
```

```
$nx=$nx+ $coor{"$m"} {"$res"} {"$r"}->x;
```

```
$ny=$ny+ $coor{"$m"} {"$res"} {"$r"}->y;
```

```
$nz=$nz+ $coor{"$m"} {"$res"} {"$r"}->z;
```

```
$ox=$ox+ $coor{"$m"} {"$res"} {"$r"}->x;
```

```
$oy=$oy+ $coor{"$m"} {"$res"} {"$r"}->y;
```

```
$oz=$oz+ $coor{"$m"} {"$res"} {"$r"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Классы

```
#!/(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } ;
```

## Основные

- Альфа-спиральные домены (218 укладок)
- Бета-структурные домены (144)
- Альфа/бета структурные домены (a/b) (136)
- (бета-альфа-бета структурные единицы)
- Альфа+бета домены (a+b) (279)
- (разделенные альфа спиральные и бета-структурные области)

```
if ($qnum) {
#system("mkdir -p $dir/$qnum");
my $filename=$ARGV[0];
$filename=$qnum.$filename;
# $filename=$chnum.".".$chnum.".".$filename;
$filename=$qnum.$chnum.$filename;
print "file: $filename\n";
open OUT,">$filename";
print OUT "Q_chain $chnum $qnum\n";
foreach my $m (sort {$a<=>$b; keys %coor}{
my %qartets = %q; # fix quartet { $coor{$m} };
my $q = $m;
#
foreach my $q { keys %qartets } { print join " ", @{$qartets{$q}} , "\n";
foreach my $s { keys %coor } {
my $sox, my $soy, my $soz;
my $sr;
foreach my $res ( @{$ $qartets{$q} } ){
#
print "$q $coor{$m} {$res} {"$R"}->x,\n";
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Классы

## Специфические

- Многодоменные белки (46)
- (сложные домены)
- Мембранные (47)
- (кроме белков иммунной системы)
- Маленькие (75)

## Формально собранные классы

- Спирализованные спирали (6)
- Низкое разрешение (24)
- Пептиды, фрагменты (116)
- Искусственные белки (42)

```
#[my %$coor,$schnum]=read_pdb($ARGV[0]);
```

```
my $sch,$schnum;
foreach my $schkey (keys %{$coor{"0"}}){my $schnum=$schkey%100;
my $sch=$schkey-$schnum;($schnum++;$sch=$ggg) };
```

```
my %qwa=find_quart($coor{"0"}); my $qnum=keys %qwa;
```

```
if ($qnum > 0){
```

```
  #system("ls -l $ARGV[1]");
```

```
  my $filename=$ARGV[1];
```

```
  $filename=$filename.pdb;
```

```
  #system("cat $filename");
```

```
  print "$filename\n";
```

```
  open OUT,">$filename.out";
```

```
  print OUT "$qnum\n";
```

```
  close OUT;
```

```
  #system("rm $filename.out");
```

# Scop Classification Statistics. 1.73 release

34 494 PDB Entries (Sep 2007). 97 178 Domains

Class	Number of folds	Number of superfamilies	Number of families
All alpha proteins	259	459	772
All beta proteins	165	331	679
Alpha and beta proteins (a/b)	141	232	736
Alpha and beta proteins (a+b)	334	448	897
Multi-domain proteins	53	53	74
Membrane and cell surface proteins	50	92	104
Small proteins	85	122	202

# Class Architecture Topology Homologous superfamily, CATH

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
my ($dir, $coor, my $schnum)=read_pdb($ARGV[0]);
my $coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $sch ){ $schnum++; $sch=$ggg } ;
```

```
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
```

```
if ($qnum > 0){
```

```
  #system("cp $dir $dir.$schnum");
```

```
  my $fdir=$dir.$schnum;
```

```
  $filename=$fdir.$schnum;
```

```
  # $filename=$fdir.$schnum.$schnum;
```

```
  # $filename=$fdir.$schnum.$schnum.$schnum;
```

```
  print "$filename";
```

```
  open OUT,">$filename";
```

```
  print OUT "#INFO: $schnum";
```

```
  foreach my $m (sort keys %{$coor{"0"}}){
```

```
    my %quartets=quartets($coor{"0"}{$m});
```

```
    my %q=find_q( $coor{"0"}{$m} );
```

```
    # foreach my $q (keys %q){
```

```
      foreach my $m1 (keys %{$coor{"0"}{$q}}){
```

```
        my $x1; my $y1; my $z1;
```

```
        my $x2; my $y2; my $z2;
```

```
        my $r;
```

```
        foreach my $res ( @{$ $quartets{$q} } ){
```

```
          # print "$q $coor{"0"}{$m1} {"$res"} {"$res"}->x,"n";
```

```
          $nx=$nx+ $coor{"0"}{$m1} {"$res"}->x;
```

```
          $ny=$ny+ $coor{"0"}{$m1} {"$res"}->y;
```

```
          $nz=$nz+ $coor{"0"}{$m1} {"$res"}->z;
```

```
          $ox=$ox+ $coor{"0"}{$m1} {"$res"}->x;
```

```
          $oy=$oy+ $coor{"0"}{$m1} {"$res"}->y;
```

```
          $oz=$oz+ $coor{"0"}{$m1} {"$res"}->z;
```

- Белок делится на домены автоматически при согласованных результатах трех алгоритмов:

- DETECTIVE (Swindells, 1995),

- PUU (Holm & Sander, 1994)

- DOMAK (Siddiqui and Barton, 1995).

- При несовпадении результатов алгоритмов – решение о доменах за экспертом

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# CATH: уровни классификации

```
my ($my $coor, my $schnum)=read_pdb($ARGV[0]);
my $my $coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch, my $schnum;
```

- Класс: основные all-alpha, all-beta, alpha-beta
- Архитектура: сходное пространственное расположение элементов вторичной структуры без учета их последовательности
- Топология (укладка): сходное взаимное расположение вдоль цепи и в пространстве элементов вторичной структуры
- Суперсемейство: предположительно или несомненно гомологичные домены
- Семейство: сходные последовательности (>35% identity и выровненные участки покрывают >60% длины)

```
foreach my $key (keys %$coor) {
    my %qwa=find_quart($coor{"0"}); my $qnum=keys %qwa;
```

```
if ($qnum > 0) {
```

```
    #system("
    my $filename=
    $filename="--s/~/V/";
    #system("
    $filename="$mdir/$filename.dat";
```

```
    print "file:
    open OUT, ">";
    print OUT "INFO: chain schnum qnum qnum qnum";
```

```
    foreach my $key (keys %$coor) {
        my %qartets= %qwa; #find_quart($coor{"$m"});
        my $sq=find($coor{"$m"});
```

```
        # foreach my $sq (keys %$qartets) { print join " ", $qartets{"$sq"}, "\n";
```

```
        foreach my $sq (keys %$qartets) { print join " ", $qartets{"$sq"}, "\n";
```

```
        my $nx=$x+ $coor{"$m"}{"$res"}{"N7"}->x;
        my $ny=$y+ $coor{"$m"}{"$res"}{"N7"}->y;
        my $nz=$z+ $coor{"$m"}{"$res"}{"N7"}->z;
```

```
        # print "$q $coor{"$m"}{"$res"}{"N7"}->x, "\n";
```

```
        $nx=$nx+ $coor{"$m"}{"$res"}{"N9"}->x;
```

```
        $ny=$ny+ $coor{"$m"}{"$res"}{"N9"}->y;
```

```
        $nz=$nz+ $coor{"$m"}{"$res"}{"N9"}->z;
```

```
        $ox=$ox+ $coor{"$m"}{"$res"}{"O6"}->x;
```

```
        $oy=$oy+ $coor{"$m"}{"$res"}{"O6"}->y;
```

```
        $oz=$oz+ $coor{"$m"}{"$res"}{"O6"}->z;
```

# Внимание

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

my ($my $coor, my $schnum)=read_pdb($ARGV[0]);
my $my $coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $sggg=substr($r,0,1); if ( $sggg ne $sch ){ $schnum++; $sch=$sggg } ;
```

```
my $my $qwa=find_quart( $coor{"0"} ); my $sqnum=keys %$qwa;
```

```
if ($sqnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
```

```
$filename =~ s/\/.*//;
my $filename=$filename . ".dat";
open $fh, ">$filename";
print $fh "#INFO Chain schnum=$schnum\n";
```

```
foreach my $m (sort { $a<=>$b } keys %coor){
my %qartets = %qwa; #find_quart( $coor{$m} );
my %q = find_q( $coor{$m} );
```

```
# foreach my $q ( keys %qartets ){ print join " ", @{$qartets{$q}}, "\n";
```

```
foreach my $q ( keys %qartets){
```

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

```
foreach my $res ( @{$qartets{$q}} ){
```

```
# print "$q $coor{$m} {$res} {"R"}->x, "\n";
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

# Классификации SCOR и CATH часто не совпадают



# Цели CASP

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

#(my %$coor,my $chnum)=read_pdb($ARGV[0]);
my %$coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $chnum;
foreach my $s (< <out keys %{$coor{"0"}>> / my $scoor=substr($s,0,1); if ( $scoor on $sch) {$chnum++; $sch=$scoor} );
my %$qwa=find_quartets($coor{"0"}); my $snum=keys %$qwa;
```

В CASP7 решались следующие вопросы:

- Насколько модели похожи на экспериментальную структуру?
- Насколько верно выравнивание?
- Были ли ранее решены подобные структуры?
- Насколько гомологичное моделирование улучшает структуру, по сравнению с копированием?
- Есть ли прогресс по сравнению с предыдущими соревнованиями?
- Какие методы самые эффективные?

```
if ($snum > 0)
#system("mkdir $ARGV[1]");
my $fname=$dir.$snum;
my $filename=$fname.".pdb";
# $filename=$fname.".pqr";
print "$filename\n";
open OUT,">$filename";
print OUT;

foreach my $m (keys %$qwa) {
my %$q;
my %$q=find_qt($coor{$m});
#
foreach my $sres (@{ $qartets{$q} }) {
print "$q $coor{$m} {$sres} {" "R" }->x,\n";
$nx=$nx+ $coor{$m} {$sres} {"N9"}->x;
$ny=$ny+ $coor{$m} {$sres} {"N9"}->y;
$nz=$nz+ $coor{$m} {$sres} {"N9"}->z;

$ox=$ox+ $coor{$m} {$sres} {"O6"}->x;
$oy=$oy+ $coor{$m} {$sres} {"O6"}->y;
$oz=$oz+ $coor{$m} {$sres} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Категории CASP

Результаты сравнивались в следующих категориях:

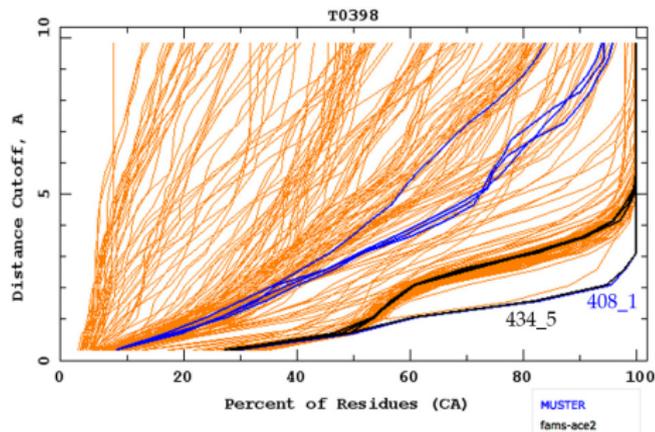
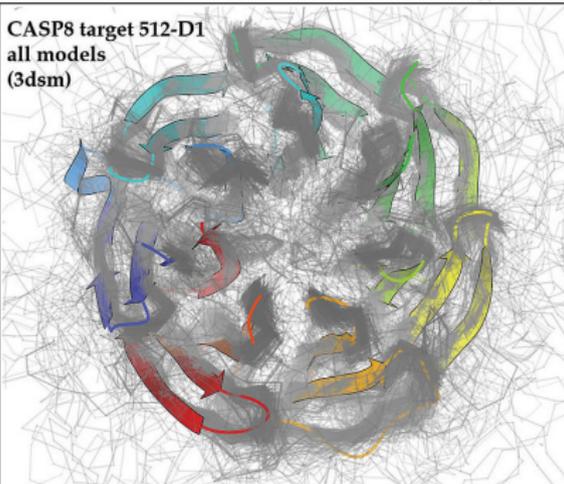
- 3D структура (все CASPs)
- Гомологичное моделирование
- Распознавание укладки.
- de novo предсказание.
- 2D структура (прекратили после CASP5)
- Предсказание структуры комплексов (CAPRI)
- Предсказание контактов (с CASP4)
- Предсказание неструктурированных областей (с CASP5)
- Предсказание границ домена (с CASP6)
- Предсказание функции (с CASP6)
- Определение качества модели (с CASP7)
- Улучшение модели (с CASP7)

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

# Визуализация CASP<sup>R</sup>

```
#!/(my %coor,my $schnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $sch){ $schnum++; $sch=$ggg } ;
```

CASP8 target 512-D1  
all models  
(3dsm)



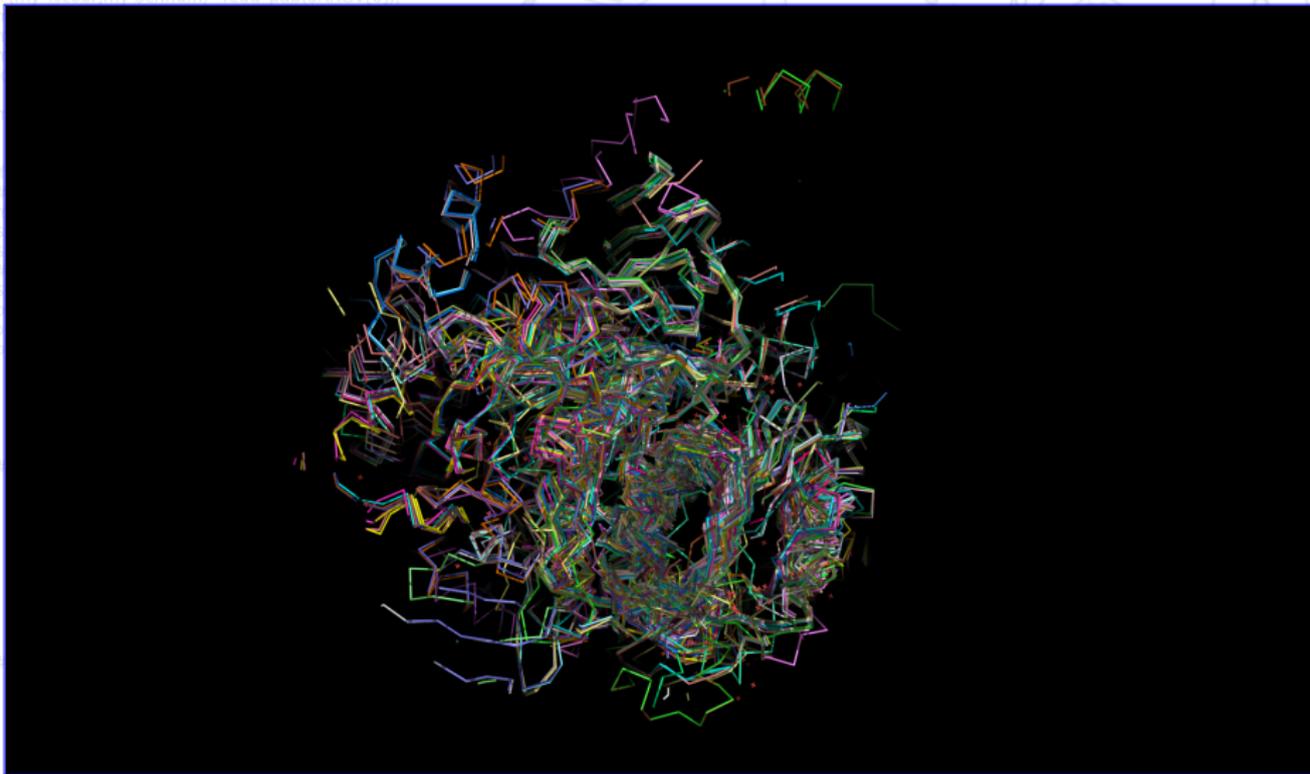
```
foreach my $res ( @{ $qartets{$q} } ){
#
print "$q $coor{$m}{$res}{"N9"}->x,"n";
$nx=$nx+ $coor{$m}{$res}{"N9"}->x;
$ny=$ny+ $coor{$m}{$res}{"N9"}->y;
$nz=$nz+ $coor{$m}{$res}{"N9"}->z;

$ox=$ox+ $coor{$m}{$res}{"O6"}->x;
$oy=$oy+ $coor{$m}{$res}{"O6"}->y;
$oz=$oz+ $coor{$m}{$res}{"O6"}->z;
```

```
#!/usr/bin/perl  
use Math::VectorReal qw( :all );
```

# Поиск ядра метилаз

```
#!/my/casper/my/Schnum) -mod_nsb($ARGV[0])
```



```
$xy=$xy+ $coor{$m}{$res}{ "O" } * $y;  
$oz=$oz+ $coor{$m}{$res}{ "O6" } * $z;
```

# Вопросы?

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){ $chnum++; $ch=$ggg } };

my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;

if ($qnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename-- s/^.*\.\.//;
$filename-- s/\.\.pdb//;
#$filename=$chnum."_"$qnum."_"$filename.".dat";
$filename="$dir"."$filename.".dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $qnum \n";

foreach my $m (sort { $a<=>$b } keys %coor){
my %qartets = %qwa; #find_quart( %coor{$m} );
my %q = find_q( %coor{$m} );

# foreach my $q ( keys %qartets){ print join " ", @{$qartets{$q}}, "\n";

foreach my $q ( keys %qartets){

my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;

foreach my $res ( @{$qartets{$q}}){

# print "$q %coor{$m}{ $res }{"$res"}->x, "\n";
$nx=$nx+ %coor{$m}{ $res }{"N9"}->x;
$ny=$ny+ %coor{$m}{ $res }{"N9"}->y;
$nz=$nz+ %coor{$m}{ $res }{"N9"}->z;

$ox=$ox+ %coor{$m}{ $res }{"O6"}->x;
$oy=$oy+ %coor{$m}{ $res }{"O6"}->y;
$oz=$oz+ %coor{$m}{ $res }{"O6"}->z;

```