分子生物第３講座/構造生物学研究センター　　2009 年 2 年生コンピュータ実習

文責　成田哲博

第十一回　pdb フォーマットと蛋白質構造

今回の目的

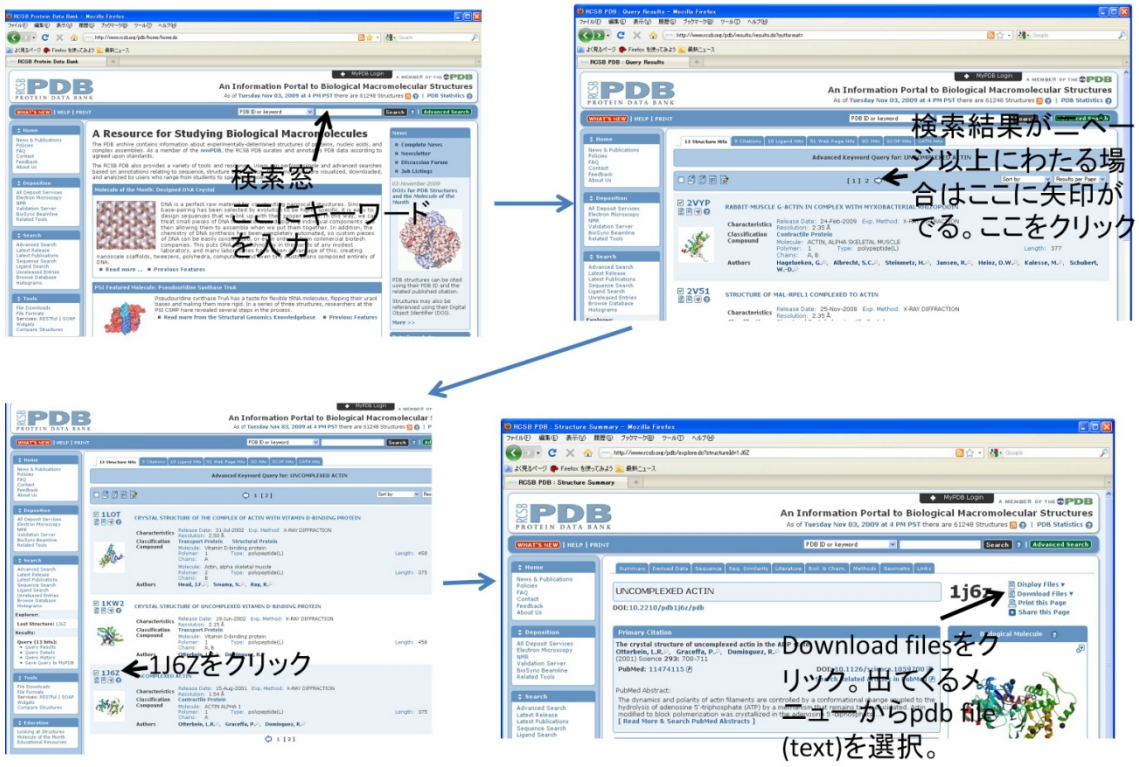
蛋白質構造を記述する pdb フォーマットの基礎がわかる。簡単な解析ができる。

Pdb database

X 線結晶解析や NMR などで解かれた生体高分子の原子座標構造を表すための共通フォーマットが策定されており、これを pdb フォーマットと呼ぶ。pdb フォーマットで書かれた原子座標構造を pdb ファイルと呼ぶ。いままで決定された構造の pdb ファイルは、

http://www.rcsb.org/pdb/home/home.do

からダウンロードできる。試しに、ウサギ骨格筋アクチン構造のひとつ、1J6Z.pdb をダウンロードしてみよう。このアドレスにアクセスすると、下のようなページになるので、矢印に示した検索窓に、uncomplexed actin と入力してみよう。すると様々な構造がでてくる。この検索結果の二ページ目に 1J6Z があるので、それをクリック。開いたページの中の Download files をクリックし、出てくるメニューから pdb file (text) を選択すると、保存場所を聞かれるので、指定すれば、ダウンロードができる。



Pdb file とは

Pdb ファイルは、厳密にフォーマットが定められたテキストファイルである。PDB ファイルの基本単位は行となっており、各行の先頭のキーワードがその行がもつ情報が何であるかを表している。Pdb ファイルには、以下のような情報が含まれている。

- HEADER: このファイルに含まれている蛋白質の総称名とこのファイルが PDB に登録された日付が書いてある。また、このデータの ID もしめされている。
- COMPND: この蛋白質の名前。
- SOURCE: この蛋白質が得られたソース（起源）。
- AUTHOR: データを PDB に登録した人の名前。
- JRNL: この蛋白質の X 線結晶構造解析の論文。
- REMARK: この構造解析に関する種々の情報。
- SEQRES: 3 文字表記のアミノ酸コードでこの蛋白質のアミノ酸配列を示す。
- FORMUL: この結晶に含まれる他の分子（低分子）に関する情報。
- HELIX: αヘリックスをとるアミノ酸の位置が残基番号で示される。βシートやターンの位置を示すためには同様に SHEET や TURN というキーワード名が使われる。
- SSBOND: 分子内の S-S 架橋の位置。
- CRYST: 格子定数、空間群および単位格子中に含まれる蛋白質分子の数。
- ORIGX および SCALE: 単位格子中で空間群の対称操作で関係づけられる蛋白質分子の座標を計算するための変換マトリックス等。
- ATOM: 各原子の座標に関するデータが入っている。各カラムの意味は以下の通り：

  カラム 1-4　ATOM
  - 7-11　Atom serial number　←原子通し番号
  - 13-16　Atom name　←原子名
  - 17　Alternate location indicator
  - 18-20　Residue name　←残基名
  - 22　Chain identification　←ポリペプチド鎖名
  - 23-26　Residue sequence number　←残基番号
  - 31-38　X (Orthogonal Å coordinates)　←原子の直交Å座標値
  - 39-46　Y (Orthogonal Å coordinates)
  - 47-54　Z (Orthogonal Å coordinates)
  - 55-60　Occupancy　←占有率
  - 61-66　Temperature factor　←温度因子
  - 73-76　ID　←識別コード
  - 77-80　Sequential number　←通し番号

  原子名は IUPAC-IUB 命名法にしたがっているが、ギリシャ文字が使えないので、α、β、γ、δ、ε、ζ、η に対しては、A、B、G、D、E、Z、H を対応させている。たとえば CA は C_α（αカーボン）を表す。また、カラム 1-4 とは、その行の 1-4 文字目という意味。
- HETATM: 蛋白質以外の原子の座標がここに示される（水分子や ATP など）。内容は全く ATOM と同じ。
- CONECT: 蛋白質分子ではない原子間の結合を表す。対になっている番号の原子間に結合があることを示す。

- TER: 一つのポリペプチド鎖の終了を示す。
- END: このファイルの終了を示す。

先ほどダウンロードした 1J6Z.pdb の中を emacs でみてみよう。まず最初に、構造の由来が書かれているヘッダー部分がある。ここには、何の分子の構造で、どの手法を用いたか、どの雑誌のどの論文に最初に載せられたかなどが書かれている。

```
HEADER    CONTRACTILE PROTEIN                     15-MAY-01   1J6Z
TITLE     UNCOMPLEXED ACTIN
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: ACTIN ALPHA 1;
COMPND   3 CHAIN: A
SOURCE    MOL_ID: 1;
SOURCE   2 ORGANISM_SCIENTIFIC: ORYCTOLAGUS CUNICULUS;
SOURCE   3 ORGANISM_COMMON: RABBIT;
SOURCE   4 OTHER_DETAILS: MUSCLE
KEYWDS    ACTIN, TETRAMETHYLRHODAMINE-5-MALEIMIDE, ADP-STATE
EXPDTA    X-RAY DIFFRACTION
AUTHOR    L.R.OTTERBEIN,P.GRACEFFA,R.DOMINGUEZ
REVDAT   2   25-MAR-03 1J6Z    1       FORMUL REMARK
REVDAT   1   15-AUG-01 1J6Z    0
JRNL        AUTH   L.R.OTTERBEIN,P.GRACEFFA,R.DOMINGUEZ
JRNL        TITL   THE CRYSTAL STRUCTURE OF UNCOMPLEXED ACTIN IN THE
JRNL        TITL 2 ADP STATE
JRNL        REF    SCIENCE                         V. 293   708 2001
JRNL        REFN   ASTM SCIEAS  US ISSN 0036-8075
```

次に続く REMARK の項には、構造の分解能、実験の詳細、結晶が持つ対称性などが書かれている。次の SEQRES の項には、この蛋白質のシークエンスが書かれている。その先にある ATOM、HETATM の項には、実際の構造データが書かれている。ATOM にはポリペプチドの座標が、HETATM は ATP や金属イオンなどのポリペプチド以外の座標が書かれている。

最初のカラムから、"ATOM"または"HETATM", 原子の通し番号、原子の名前、残基の名前、分子の ID, 残基番号、その原子の x 座標、y 座標、z 座標、占有率、温度因子がかかれている。このファイルでは、最後に原子の種類が書かれているが、これはオプションである。

たとえば、このファイルの ATOM の項の二行目

```
ATOM      2  CA  GLU A   4      -14.630  11.239   6.884  1.00 45.46           C
```

は、このファイルの二番目の原子で、Cα原子、グルタミン酸、chainIDはA、残基番号は4, 座標は、(-14.630, 11.239, 6.884) (単位はÅ)であることを表している。占有率と温度因子は、結

晶学特有のパラメータなので、今回は省略する。

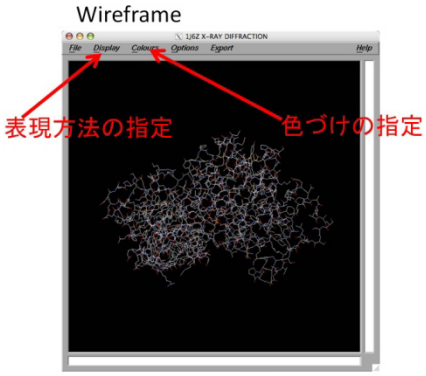
Pdb viewer rasmol

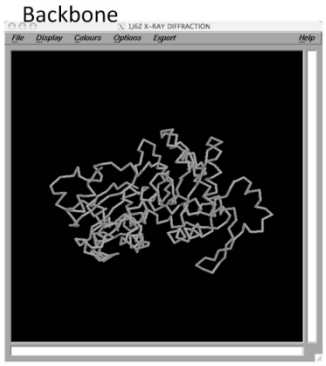
Pdb ファイルを三次元的に見るプログラムは多数あるが、もっとも歴史がありどの OS でも動くプログラムが rasmol である。rasmol の使い方は簡単で、

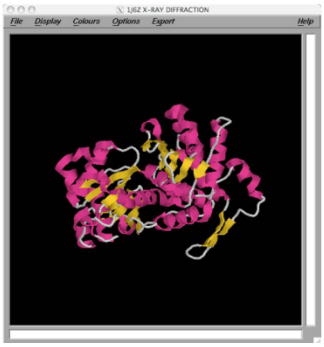
% rasmol pdbfilename

で起動する。たとえば、先ほどの 1J6Z.pdb であれば、

% rasmol 1J6Z.pdb







Display: Cartoons
Colours: Structure

である。最初はワイヤーフレームと呼ばれる形式で表示されていると思う。様々な表現方法を指定できる。表現方法は、ウィンドウ上のメニューの Display で選べる。Backbone や Cartoons などにしてみよう。また、色のつけかたはメニューの Colours で選べる。また、画面の中で左ボタンを押しながら、マウスを動かすと構造が回転する。右ボタンを押しながら動かすと、平行移動する。

一方、rasmol を起動した xterm の上には、

RasMol>

という表示がでている。これは rasmol のプロンプトで、ここにいろいろなコマンドを打ち込むことができる。また、rasmol からの情報もここに表示される。たとえば、画面の中の蛋白質構造の上をクリックすると、

Atom: CA 1179　Group: GLY 156　Chain: A

のような表示が現れる。これは、クリックした原子が Cα で、1179 番目の原子。残基番号 156 のグリシンで、Chain ID が A であることを示している。Chain ID は、pdb ファイルの中に複数のポリペプチド鎖が含まれている場合、何番目のポリペプチド鎖かを指定する。コマンドプロンプトから、蛋白質の特定の部分を指定して、そこだけ表現方法を変えたり、色を変えたりすることもできる。たとえば、Chain Id が A の残基番号 200 の残基を指定するには、rasmol のプロンプトから、select 200a とすれば良い。

RasMol> select 200a

11 atoms selected!

この上で、rasmol のメニュー上の Display から Spacefill を指定すると、残基番号 200 のアミノ酸残基だけが Spacefill モデル(空間充填モデル)の表現に変化する。もし、Chain ID A の 20-50 番目の残基を一気に指定したければ、select 20-50a とする。

RasMol> select 20-50a

226 atoms selected!

この領域の色を変えたければ、コマンド color を使う。

RasMol> color red

そうすると、20-50a の領域が赤に変化する。また、select は,で区切って複数の領域を指定できる。たとえば

RasMol> select 20-50a, 200a, 300-320a,350a

とすれば、chainID A の 20-50,200,300-320,350 の残基を指定することになる。

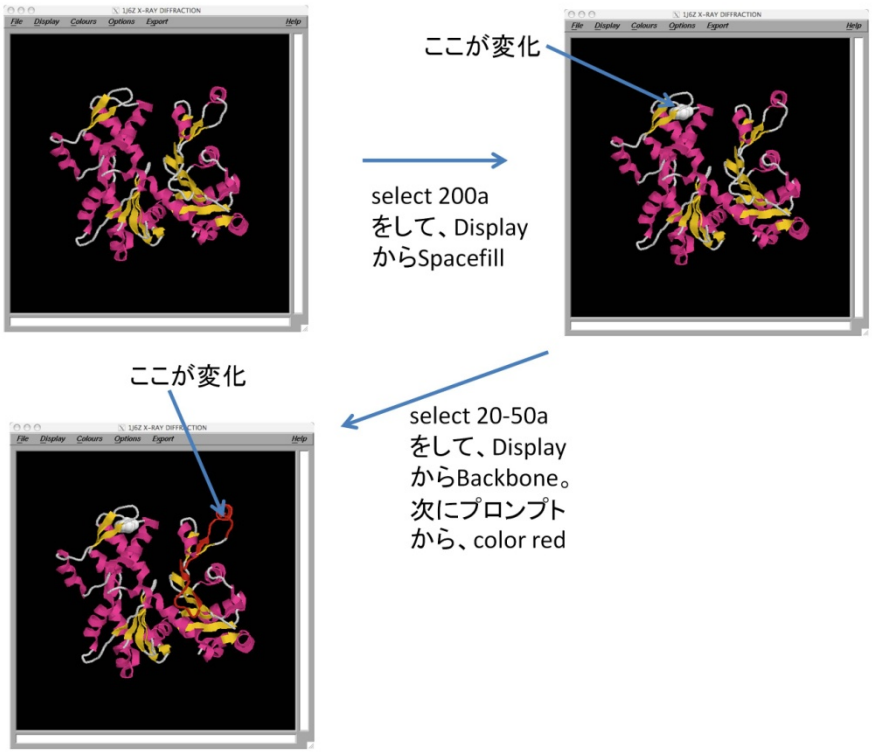
ズームもできる。そのためには、ズームする中心の残基を center で指定し、コマンド zoom を用いる。zoom においては、100 が等倍である。たとえば、157a を中心に、二倍に拡大したければ、

RasMol> center 157a

RasMol> zoom 200

となる。また、zoom や水平移動、回転状態を最初の状態に戻したければコマンド reset を用いる

RasMol> reset



Rasmol のマニュアルは
http://www.rasmol.org/software/RasMol_2.7.5_Manual.html
にある。これを参考にいろいろ自分で試してみよう。

複数のポリペプチド鎖がある場合

複数のポリペプチド鎖がある構造の例として、講義資料のページから、F-actin 構造をクリック、factinn6.pdb をダウンロードしてみよう。これは、アクチンが重合したときの構造を示している。このファイルには 6 分子のアクチンが含まれており、chainID はそれぞれの分子に B,C,D,E,F,G がわりあてられている。rasmol で開いてみよう。Rasmol のメニューの Colours から、Chain を指定すると、分子ごとに別の色がわりあてられ、見やすくなる。

Select いろいろ

ここで、select についてもう少し見てみよう。chainID を省略すると、全てのポリペプチド鎖に対する指定になる。

```
RasMol> select 10
RasMol> color green
```

とすると、全てのポリペプチド鎖の残基番号 10 の残基が緑色になる。chainID C のポリペ

プチド鎖の残基全てを指定したければ、

RasMol> select *c

とする。ファイル内の全ての原子を指定したければ、

RasMol> select all

でよい。

課題１　rasmol の習得

factinn6.pdb の中の分子 D の N 末端と C 末端の残基を白の spacefill で表示せよ。その場所がわかりやすいような図を作り、レポートせよ。

Pdb file の操作

テキストエディタによる操作

Pdb ファイルはテキストファイルにすぎないので、中を直接 emacs などのテキストエディタで直接いじることができる。簡単な例を見てみよう。emacs で 1J6Z.pdb の ATOM 項最初の行

```
ATOM      1  N   GLU A   4     -16.021  11.749   6.951  1.00 46.20           N
```

を

```
ATOM      1  N   GLU A   4     -56.021  11.749   6.951  1.00 46.20           N
```

に変えて、1J6Z2.pdb として保存、rasmol で開き、spacefill で表示してみよう。原子が一個だけ離れているのが見えるはずである。離れた原子をクリックすると確かに残基番号 4 の原子 N であることがわかる。この座標を適当に変えてどう動くか試してみよう。

Python による Cα原子の抜き出しと find メソッド

テキストファイルである以上、python による操作は簡単である。試しに、pdb ファイルから Cα原子だけを抜き出すプログラム caonly.py を書いてみよう。原子名はカラム 13-16、つまり 13 から 16 文字目に書かれている。従って、13-16 文字目をスライスによって抜き出すことで、簡単に原子名を抜き出すことができる。また、このプログラムのために使うと便利な文字列用メソッド find を紹介しよう。

変数名.find(‘探す文字列’)

とすると、変数の中で、’探す文字列’がスタートする index を返す。存在しなければ-1 を返す。Python の対話モードで見てみよう。

```
>>> a="abcdef"
>>> a.find('bc')
1   #’bc’は index 1(二文字目)からスタートする。
>>> a.find('ef')
```

4　#'ef'は index 4(五文字目)からスタートする。

```
>>> a.find('z')
```

-1　#'z'は a の中には存在しない。

これを使って、caonly.py を書くと以下のようになる。

caonly.py

```
#!/usr/bin/env python

import sys
infile=sys.argv[1] #infile:読み込むpdbファイル
outfile=sys.argv[2] #outfile:書き出すpdbファイル
fin=open(infile,'r')
fout=open(outfile,'w')

for line in fin:
    head=line[0:6]
    if head.find('ATOM') != -1:  #1文字目から6文字目まで(head)の間にATOMが含まれているか。Cαはの項の中だけから探せば良い。
        atomname=line[12:16]  #13から16文字目を取り出す。
        if atomname.find('CA') != -1:  #取り出した中にCAが含まれていれば
            fout.write(line)   #foutに書き出す。
fin.close
fout.close
```

実際に

```
% caonly.py 1J6Z.pdb 1J6ZCa.pdb
```

として、その結果が Cα原子しか含まれていないことを確かめてみよう。

課題２特定 Chain ID の抜き出し

Pdb ファイルから特定の chain ID の原子だけを抜き出すプログラム singlechain.py を書け。それを用いて、実際に factinn6.pdb から chainID が D の原子だけ抜き出し、その結果もレポートせよ。

Python による分子間インタフェースの同定

ここまでで見たように、python を使えば pdb ファイルをわりと簡単に変更したり解析したりできる。ここでは少し実用的なプログラムとして、pdb ファイルから分子間の結合部位

を取り出すプログラム pdbinterface.py を書いてみよう。

```
% pdbinterface.py pdbファイル名 chainID1 chainID2 threshold  output
```

とすると、pdbファイルの中のchainID1で示される分子のある残基Aのなかのどれかの原子が、chainID2で示される分子のある残基Bの中のどれかの原子とthreshold(Å)以内の距離に存在する場合、AとBの二つの残基は、結合部位に存在すると判断する。そのような残基のリストをoutputに出力する。たとえば、

```
# ./pdbinterface.py factinn6.pdb C E 4  CEinterface.txt
```

とすると、factinn6.pdbの中で、C分子とE分子の結合部位にある残基を4ÅのthresholdでSearchするという意味になる。

pdbinterface.py

```
#!/usr/bin/python

import sys

infile=sys.argv[1] #pdb file名
mol1=sys.argv[2] #分子1のID
mol2=sys.argv[3] #分子2のID
th=sys.argv[4]  #threshold
out=sys.argv[5] #出力ファイル

th=float(th)*float(th)  #計算の都合で、thresholdを二乗にする。
f=open(infile,'r')
x=[]
y=[]
z=[]
res=[]
mol=[]
flag={} #結合部位に存在する残基名を保存するディレクショナリ

for line in f: #全ての原子のx座標、y座標、z座標、残基番号(res)、chainID(mol)をリストに保存する。
    name=line[0:6]
    if (name.find('ATOM') != -1):
        x.append(float(line[30:38]))
        y.append(float(line[38:46]))
```

```
        z.append(float(line[46:54]))
        res.append(line[22:26])
        mol.append(line[21:22])
f.close
```

```
for n in range(0,len(x)):
    for m in range (n+1,len(x)): #保存された全ての原子のペアについてループする。
        if ((mol[n].find(mol1) != -1 and mol[m].find(mol2) != -1) or (mol[n].find(mol2) != -1 and mol[m].find(mol1) != -1)): #上の行とこの行の間には改行なし。あわせて一行。
```

もし、mol[n]とmol[m]が、mol1, mol2またはmol2, mol1の組み合わせであれば、結合部位かどうかを判定する。

```
            d = (x[n]-x[m])**2 +(y[n]-y[m])**2 + (z[n]-z[m])**2
            if (d < th):  #原子間の距離の二乗を計算し、thより小さいかを判定。
                resmol1=res[n]+mol[n] #小さければ結合部位と判断。
                resmol2=res[m]+mol[m]
                flag[resmol1]=1
                flag[resmol2]=1
```

結合部位であれば、flag[残基番号+chainID]に1を代入する。こうすることで、flag[残基番号+chainID]が生成される。

```
f=open(out,'w')
for key in flag.keys():
    f.write(key+"¥n")
f.close
```

#存在する flag のキーを出力する。

実際に

./pdbinterface.py factinn6.pdb C E 4 CEinterface.txt

を実行すると、CEinterface.txt は以下のようになる。

```
40C
  42C
 244C
 291E
 287E
 242C
 323E
 169E
```

```
  39C
 167E
 205C
  62C
 171E
 204C
 325E
  45C
  41C
 …
```

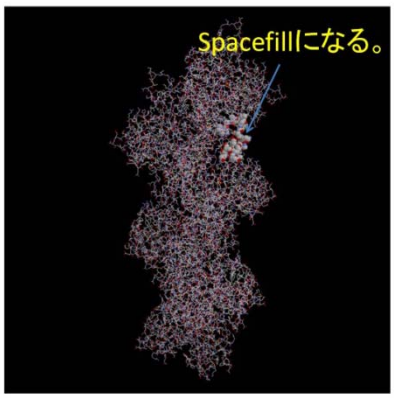
Rasmol のスクリプトと、残基リストの表現

これだとわかりにくいので、rasmol上で見られるようにしよう。rasmolには、コマンドを羅列したスクリプトを読み込むことができる。たとえば、以下のようなtest.rasを作ってみよう。

```
select 40-60E
spacefill
```

これは、40-60E を spacefill で表示するという意味である。
その上で factinn6.pdb を rasmol で開き、test.ras を読み込んでみよう。rasmol におけるスクリプトの読み込みは、source コマンドを用いる。

```
% rasmol factinn6.pdb
RasMol> source test.ras
```



こうすることで、図のように 40-60E が spacefill 表示になることを確かめよう。
結合部位のリストに関して、このようなスクリプトを書いて spacefill で表示させるようにしたらわかりやすいだろう。そのようなスクリプト makespacefill.py は簡単に書ける。
makespacefill.py 入力ファイル　出力ファイル
入力は pdbinterface.py の結果ファイル、出力ファイルは rasmol のスクリプトで、結合部位の残基を spacefill で表示させる。

makespacefill.py

```
#!/usr/bin/python
```

```
import sys
infile=sys.argv[1]
outfile=sys.argv[2]

fin=open(infile,'r')
fout=open(outfile,'w')

for line in fin:
    fout.write('select '+line)
    fout.write('spacefill'+'¥n')
fout.close
fin.close
```

このプログラムを使って、先ほどの CEinterface.txt **をスクリプトに変換しよう。**

% makespacefill.py CEinterface.txt CEinterface.ras

あとは、

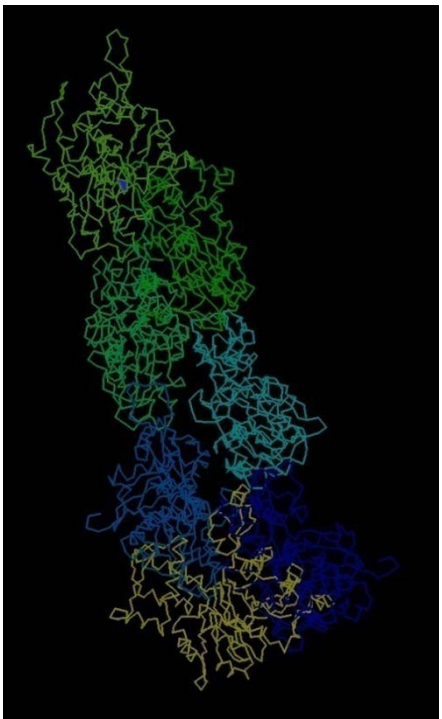
% rasmol factinn6.pdb

RasMol> source CEinterface.ras

とすると、分子 C と E の結合部位に存在する残基が spacefill で表示される。

Structural alignment

似た二つの蛋白質構造を比べてみたい場合は良くある。今回はアクチンフィラメント構造である factinn6.pdb の中のアクチン分子と、単量体アクチンである 1J6Z.pdb のアクチン分子の間の構造変化を調べてみよう。二つの分子間の構造変化を調べるためにはまずは、二つの構造を重ねて表示しなくてはならない。そのために一番簡単な方法は、二つの pdb ファイルを結合することである。cat を使うと二つのファイルを結合できる。



cat ファイル名１　ファイル名２　…　> 出力ファイル名

のようにすると、いくつのテキストファイルでも一つのテキストファイルにまとめることができる。たとえば、

% cat 1J6Z.pdb factinn6.pdb > factinn6.1J6Zadd.pdb

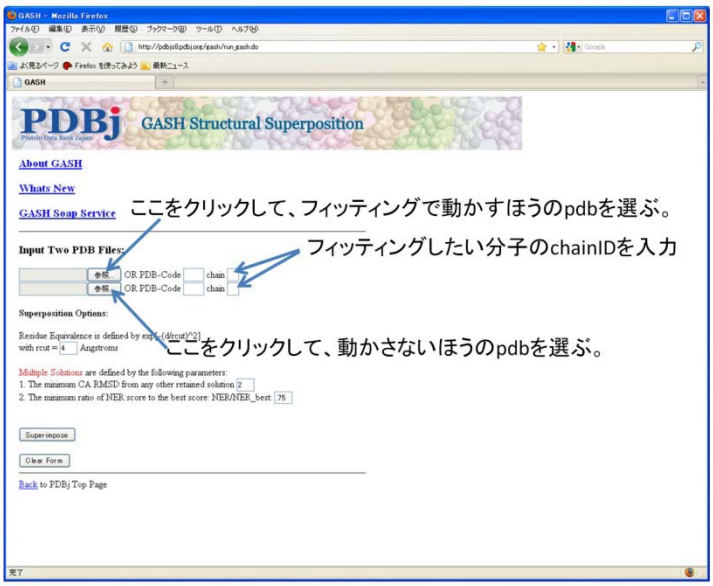
とすると、1J6Z.pdb と factinn6.pdb の内容が factinn6.1J6Zadd.pdb の中にコピーされる。まずはこ

れを rasmol で表示してみよう。右図は、メニューの colour→Chain で色づけしてある。黄色が chainA つまり 1J6Z.pdb であるが、ほかの factinn6.pdb の分子と位置がまったくあっていない。比較するためには、比較したい分子同士をまず同じ位置と方向にそろえなければならない。この位置と方向を合わせることを structral alignment と呼ぶ。

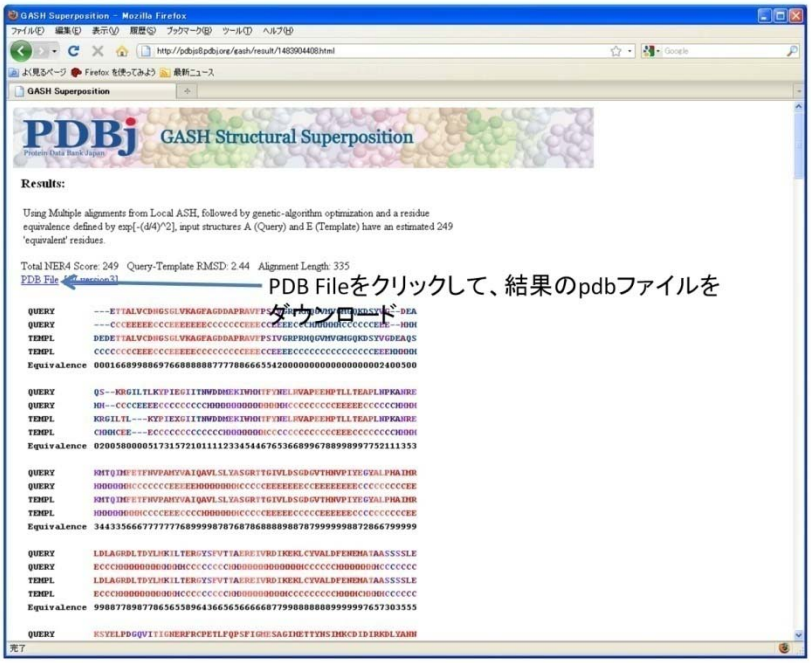
様々なソフトウェアが structural alignment をサポートしているが、今回は web 上で使える PDBj の GASH というアラインメントを試してみよう。使い方は簡単である。

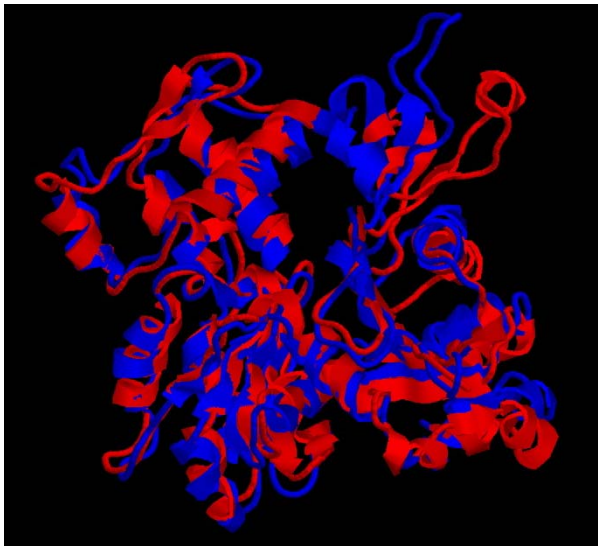
http://pdbjs8.pdbj.org/gash/run_gash.do

にアクセスすると、右図のようになる。上側は、アラインメントで動かすほうの pdb ファイル、下側はアラインメントで動かさないほうの pdb ファイルである。上側の構造を下側の構造に合わせてくれる。しばらく待つと次ページのような画面になる。いくつかの候補が表示されるが、上にでてくるほどスコアが高いので、今回は、一番上の結果の pdb ファイルをダウンロードしよう。



結果のファイルには、フィッティングで動かしたほうの構造の chainID は B で、動かさないほうの chainID は A になっている。Rasmol で表示すると以下のようになる。このとき、メニューから Display で Cartoons を選び、Colour から Chain を選んで表示した。青が chain A、赤が chain B である。二つの分子の位置がちゃんと合っているのがわかる。





課題３　GASH を実際に使う。

GASH を用いて、factinn6.pdb の分子 E に対して、1J6Z.pdb の位置と方向を合わせよ。その結果を rasmol を使って表示し、二つの位置があっていることを示せ。

構造の比較

Structural alignment が終わったら、いよいよ構造変化部位を同定しよう。今回の場合は同じ蛋白質の違う構造を比べるので、アミノ酸シークエンスは同じである。Cα原子だけを比べることにする。そのためのアルゴリズムの例としては以下の通り。入力パラメータは、pdb ファイルと、比べる分子の chainID(二つ)、それ以上位置にずれがある場合に構造変化していると決定するための閾値である。pdbinterface.py が参考になるだろう。

1 全ての Cα の座標とその chainID、残基番号をリストに保存する。

2　比べる分子間で、残基番号が同じ物同士で位置を比べ、そのずれが threshold 以上であれば、その残基名を保存する。

３最後に 2 で保存された全残基名を出力する。

課題４　構造変化部位の同定

実際に、構造変化部位を同定するプログラム structchange.py を書き、課題３の GASH の結果に対して実行することで、アクチンが単量体からフィラメント構造になるときに変化が大きい部位を決定せよ。また、その部位を rasmol を用いてわかるように表示せよ。