

Übung: Protein Function Prediction

WS 2007/08 Übung zu Anwendungen der Strukturanalyse (00.904)

11-Dec-07

1. Addresses	1
2. Gathering sequences.....	1
2.1. Collection of your PDB sequence	1
2.2. Generation of random sequences	2
2.3. Reordered sequence	2
2.4. Pseudo-random sequence with bias	3
2.5. Write your own random sequence generator	3
3. Using the sequences	4
3.1. A sequence motif finder	4
3.2. A neural network-based predictor	4
4. Assignment.....	5

In the lectures, we concentrated on protein structure prediction, mostly looking for homology or patterns in sequences or structures. Here, we will test some web services for prediction based solely on sequence. We can divide this Übung into two parts

1. Collection of sequences and
2. Collecting results from servers

Most of the time will be spent on making the sequences and writing a program to generate random sequences (section 2.5 on page 3). The report should be handed in by 10. Jan 2007.

1. Addresses

Protein data bank (PDB): www.rscb.org

Protfun server: www.cbs.dtu.dk/services/ProtFun/

Prosites: www.expasy.org/prosite/

2. Gathering sequences

2.1. Collection of your PDB sequence

You have to find a protein sequence based on your name. The name of the Bundeskanzlerin is Angela Dorothea Merkel. Her initials are ADM. Her first protein would be 1adm which happens to be an adenine-N6-DNA-methyltransferase.

Do the same with your name. If you do not have a middle name, borrow one from your neighbour or the Bundeskanzlerin.

Visit the protein databank (PDB) and look up the protein with your name. Find the number of residues in the different chains (Click on the tab near the top of the page called "Sequence Details") Now, we want to make sure that you have an interesting sequence, so

while (you have selected a protein with < 60 residues)
go to next alphabetical protein (1abc -> 1abd or 1xyz->1xza)

You should now have a real protein from the PDB with more than 60 residues.

Go back to the initial page for the protein. On the left of the page, click on "FASTA sequence" and store the sequence in a filename like 1adm.fasta. You will need this later. Go down to the bottom of the page and find "GO Terms". These are gene ontology terms. Note these down.

2.2. Generation of random sequences

We would expect real proteins to have some function and proteins from the protein data bank to have a known function. It is more fun to see what functions are given to random, non-existent proteins. We will use three kinds of random sequence

1. the sequence of the protein from your name, but re-ordered
2. quasi-random sequence with correct amino acid composition
3. quasi-random sequence generated by a program written by you

2.3. Reordered sequence

Take the protein sequence which you obtained from the protein data bank. If your sequence is in a file with a name like, 1adm.fasta, then

- * edit the file 1adm.fasta and remove any comment lines which begin with a ">" character.
- * save the file in a file with a name like 1adm_nocomment.fasta

Remember the name 1adm is from Angela Merkel. Your file will have a different name. Now, make a scrambled version of the sequence with a command like¹

```
~torda/bin/shuffle_seq.x 1 "`cat 1adm_nocomment.fasta`"
```

Be careful with the double quote (") characters and backticks (`). They are all necessary. This command should print a pseudo-randomised version of the input sequence. If it works, then do this again, but save the output to a file with a command line and generate 5 sequences

```
~torda/bin/shuffle_seq.x 5 "`cat 1adm_nocomment.fasta`" >  
1adm_reorder.fasta
```

The greater than sign (>) sends the output to a file. The command should be on one line, not split in two as typed here.

Edit the file you created.

- * remove the first sequence labelled "> input"

¹ Program originally written by Gundolf Schenk

* split the sequences into five separate sequence files. Use a name you can remember like 1adm_reorder1.fasta, 1adm_reorder2.fasta, ...

2.4. Pseudo-random sequence with bias

The sequence you just generated has the composition of a real protein, but the amino acids are in the wrong order. Now, you should generate a sequence which is random, but probably has a realistic amino acid content. For example, real proteins typically have about 1.5 % tryptophan and about 8 % alanine. This means we would like pseudo-random sequences to have about 1.5 % trp and 8 % ala. We use a simple perl script for this. The name is

~torda/bin/rand_aa.pl

The command line arguments can be seen by typing

perldoc ~torda/bin/rand_aa.pl

This program is deterministic. It always generates the same output for a given random seed. Now we need a random seed. Take your first name (for example "Angela"). Turn the first three characters into numbers (a=1, b=2, ..) and add them together. Angela Merkel would use

$$a + n + g = 1 + 14 + 7 = 22$$

so the Bundeskanzlerin would use a seed of 22. In order to generate a biased, quasi-random sequence based on her name, she would type

~torda/bin/rand_aa.pl -s 22 120 > rand1.fasta

To generate a different sequence, the seed (22) should be set to a new number. A length of 120 residues is a typical protein size.

Generate about 5 sequences with names like rand1.fasta, rand2.fasta, rand3.fasta... Use successive random seeds. If you like shell scripting, and you use bash as your shell, you would write

seed=22

for i in 0 1 2 3 4 ; do

a=`expr \$seed + \$i`

~schenk/teaching/AST/03/bin/rand_aa.pl -s \$a 120 rand\${a}.fasta

done

If you use a different shell, your command would be different or just run the command by hand. Your random seed is probably different from the Bundeskanzlerin's (22). You should now have five pseudo-random sequences, generated with appropriate bias and stored in names you can find later.

2.5. Write your own random sequence generator

You may use any programming language you like. The interface can be very simple like

your_prog nn

where your_prog is the name you give to your program and nn is the length of the random sequence. Regardless of the programming language, you will need something like this pseudocode:

```

length := nn from command line
set the random number seed
for (i = 0; i < length; i++)
    r := random_number from 0 to 19
    c := amino acid corresponding to r
    add c to output

```

You can make the program as sophisticated as you like.

Your program must be deterministic. This means the seed for the random number generator must be controlled by you. In perl, you might use **srand(seed)** to set the seed and **int(rand(20))** to return a number between 0 and 19. In C, you could **srand48(seed)** to set the seed and **mrnd48()%20** to get a number between 0 and 19. Ruby users might use **srand seed** and **rand(20)**.

Ideally, you should read the seed and sequence length from the command line (**(argc, argv[])** in C or **@ARGV[]** in perl). If this is your first venture into programming, then keep the task simple and fix some constants in your code.

Finally, use the program and generate some random sequences (5 or 10).

3. Using the sequences

Before starting this section, make sure you have stored sequences for

- * the protein based on your name
- * pseudo-randomised sequences based on the protein from your name
- * pseudo-random sequences based from the generator with correct amino acid composition
- * pseudo-random sequences from your own generator

3.1. A sequence motif finder

For this server, you will need to copy and paste your sequences.

Visit the prosites server.

For each sequence, paste your sequence into the box near the bottom of the page and scan for motifs.

Make a note

- * where the sequence came from (real, shuffled, biased-random, random)
- * how many motifs were found and, for each motif:
 - * the motif name
 - * consensus sequence

3.2. A neural network-based predictor

In the lectures, we discussed looking for motifs in sequences. The prosites server uses this idea quite explicitly. One may also suspect there are trends due to composition or size of sequences. This would then be a candidate for a neural network based predictor, where one does not know the rules in advance, nor the relative importance of the factors. The Danish protfun server is based on this philosophy. It is interesting to see whether it finds the correct function for a real protein and

how it responds to fictitious proteins with sequences that do not really exist. With the protfun server, you can upload sequences from files, so there is no copying and pasting involved.

Visit the profun server.

Upload each sequence and collect the predictions.

Make a note

- * where the sequence came from (real, shuffled, biased-random, random)
- * Note
 - * the most likely functional category and the associated probability
 - * the enzyme/non-enzyme probabilities
 - * the most likely gene ontology category

For your real protein, check to see if the prediction agrees with any information on the PDB web page for the protein.

4. Assignment

1. Mail your program (source code) to **torda@zbh.uni-hamburg.de**
2. Write your name including your middle name and the random seed you used for the biased random sequence generator.
3. Write the name of your real protein (from the protein data bank)
4. For the "prints" server
 - * if it predicted motifs in your real sequence, is there any evidence from the PDB web page that they may be correct ?
 - * For the randomised sequences, are there any motifs which are most often predicted ?
5. For the "protfun" server
 - * if your real (PDB) protein had gene ontology (GO) keywords, compare these to the predicted ones
 - * survey the predictions for pseudo-random proteins. Are there any enzyme classes or gene ontology keywords which are more often predicted than others ?
6. Imagine a world with only two kinds of function. 80 % of the molecules have function 1 and 20 % have function 2.
 - * In the absence of any further information, what is the best prediction you can make for a new molecule ?
 - * If you see a bias in the results from the protfun server, is this bad ? Answer in terms of what you wrote for the previous question.
7. Imagine you want to build a web server which detects motifs in proteins. You, however, would like to give an estimate of confidence to your predictions. How would you estimate the probability of a prediction being seen by chance ? In order to calibrate your server, you have lots of computer time and a reliable pseudo-random protein sequence generator.