Übung: Breakdown of sequence homology

WS 2010/11 Übung for AST

Assignment due: 24 Jan 2011

Exercise on 10 & 17. Jan 2011

In the lectures, we talked about sequence homology (sequence similarity). The question here is where simple homology detection breaks down. If you have a sequence, do you expect to reliably find homologues with 90 % or even 20 % sequence similarity ? Unfortunately, much of bioinformatics is empirical. Many beliefs come from observation rather than rigorous theory. You can simulate searching for remote homologues by gradually changing the residues in a sequence. If you take the sequence for protein X from a databank and search in a databank, you should find protein X. If you change 10 % of the residues, it is as if you have a 90 % sequence identical homologue. If you change more of the residues, you simulate searching with more remote homologues.

There is a small amount of programming in this Übung and, in general, there should be *n* different programs from *n* different students. If you have not programmed before, write this in your report and use the program from a friend.

1. Addresses

Fasta server: `fasta.bioch.virginia.edu`

2. Gathering sequences

2.1. Collection of your PDB sequence

As in a previous Übung, you have to find your personal protein. This time, you must use the initials of your father or mother. The father of the Bundeskanzlerin was Horst (H) Kasner (K) Merkel (M). When the Bundeskanzlerin does this Übung, she would look for a protein 1hkm which happens to be a chitinase enzyme from humans.

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") 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. Near the top, you should see a tab called "Sequence". From there, there should be a link to download the sequence in fasta format. If the protein has more than one chain, download the first chain.
Store the sequence in a filename like 1kdm.fasta. You will need this later.

2.2. Your random seed

You must have your own seed for random numbers. Take the year of your birth, your father's year of birth and your mother's year of birth and add them together. Angela Merkel (born 1954) would use 1954 + 1923 + 1925 = 5802 as her random seed.

2.3. Generation of random sequences

You must write a program which can

- read an amino acid sequence from a file
- read a number from the command line - the fraction of the amino acid sequence to randomise
- scramble (re-order) a fraction of the amino acid sequence
- write the new sequence to a new file

This will change the sequence, but not the amino acid composition. The random sequences will be very protein-like. You can write in any language you like. If you produce a working program in

fortran77, you automatically receive a beer. If you produce a working program in assembler, you get a bottle of sekt.

Your program must control its random number seed. To save time, you can hard code this. If you have more time, you can make this a command line argument.

2.3.1. Necessary functions

- Two kinds of random numbers

1. Return a floating point number between 0 and 1.

2. Return an integer between 1 and a number *N*

- A function to compare two sequences and measure sequence identity

In pseudo code, this is very easy, but the details will depend on the language and how it represents strings. It should look something like:

```
float
identity (sequence1, sequence2, n)
     integer count = 0
     for (i=0; i < n; i++)
          if (sequence1[i] == sequence2[i])
               count++
     return ( (float) count / n)
```

Be careful with this simple function. You need the `(float)` cast in C, otherwise you will get integer division and truncation.

- The randomizing function

We do not need to change an exact number of residues. We can do this in a probabilistic manner. If we are going to change 10 % of the residues, there is a 0.1 probability of a single residue changing. We are going to work by interchanging residues, so each swap will affect two residues. We begin by taking the probability of a change and dividing it by two. Say `frac` is the fraction of the sequence to randomise. In C-like pseudocode, the idea would be:

```
char *
randomize (sequence1, sequence2, n, frac) {
     frac /= 2.0;
     set sequence2 = sequence1;
     for (i = 0; i < n; i++) {
          int k = i;
          while (k == i) {    /* find the position to swap with */
               k = random (0..n);           /* a random integer */
          }
          r = rand (0..1)          /* a random float from 0 to 1 */
          if (frac >= r) {
               tmp = sequence1[i];
               sequence1[i] = sequence2[k];
               sequence2[k] = tmp;
          }
     }
     return sequence2;
}
```

Note that we have a `while` loop so that we do not try to swap an amino acid with itself. At the top of the function, we divide `frac` by 2. If we want to change 30 % of the residues, we try to do swaps at 15 % of the positions. The details are very language dependent. In C, you have to pass in the size `n` of the array, but in perl this is not necessary. In C, it is easiest to use array notation, but in perl, you could use the `substr()` function. Be careful with storage. In C, it is easiest for the caller to allocate space for the string (in `sequence2`). In languages with garbage collection you could build `sequence2` and just return it.

- File reading and writing

A FASTA sequence file from the protein data bank contains a comment line at the start. This is not part of the sequence. Your program must recognize this. An elegant approach is to read the first line of the sequence file and ignore the first line if it begins with the ">" character.

When writing the output, you can put the comment line to good use. After you have calculated the sequence identity, write it into the comment line. For example, you may have wanted to change 15 % of the residues, so one expects the sequence identity to be near 85 %. Because the method is crude, you may have 86.343434 % sequence identity with the input. Your output file should begin with a line like

```
> 0.863 sequence from xxxx.seq NN residues
```

where `xxxx.seq` should be the input filename and `NN` should be the number of residues in your sequence. Do not give more than 3 significant figures in the sequence identity.

3. The final program

A simple interface to the program would be like:

```
random_seq  infile outfile fraction seed
```

where

`infile` is the name of the file containing the sequence to be randomised

`outfile` is the output file name

`fraction` is a floating point number saying how much of the sequence should be randomised. 0.10 would ask the program to change 10 % of the residues

`seed` is the seed for the random number generator (an integer)

The final program should look like

```
int
main (argc, argv)
     read_command_line_arguments
     set_random_seed
     read_input from infile
     randomise
     calculate sequence identity
     write the sequence identity to standard output
     write new sequence to outfile
}
```

4. Check your program

Do the following checks.

- if there is an error on opening the input file, the program should stop gracefully
- if the input has *N* residues, the output should have *N* residues
- If your input sequence is short like "ABCDEFGHIJ" and you change 100 % of the residues, you can see in the output if most of the residues have changed, but all the letters from A to J are present.
- if you ask the program to change 10 % of the residues, you should have close to 90 % sequence identity with the input. On large sequences, the error should not be too large
- if you ask the program to change 100 % of the residues, you would not expect 0 % sequence identity.

5. Sequence searches

Our input sequence came from the protein databank so we will use fast sequences against the protein databank sequences. One could use web interface to "blast", but we will use "fasta". It is fast enough and has slightly better statistics.

Visit the fasta web site. Follow the link to "Protein-protein FASTA".
Under "(C) Database" change the option to "NCBI PDB structures".
Near the bottom of the page is a box labelled "E():". Put in a number like 1e-5 (which means 1×10^{-5}). This is just a filter to reduce the output and ask the program not to print out sequences with larger *e-values*.
Paste in your original sequence with no changes and "Search Database". The page should show you high scoring alignments. Check very carefully the name of the best scoring protein. It should be 100 % identical, to your input, but it may not have the same name. This is not an error. Sometimes, the same protein structure is solved several times and appears in the protein databank under different names. The database used by the "fasta" server only tells you the first name.

Now generate a sequence with about 40 % of the residues changed and repeat the sequence search. Remember to specify the database. This search should also find your original sequence in first rank. Note down the *e-value* (the number of times you would expect to see this observation by chance). It is probably a very small positive number. Note down the sequence identity. Count the number of hits with the *e-value* better than the threshold.

6. Assignment

For each student, there should be one assignment and program handed in. Please hand in your answers to the questions as a .pdf or .txt file.

1. Mail your program (source code) to ast_uebung@zbh.uni-hamburg.de. Send in your program as plain text or a tar archive. Do not hide it in html or xml.

2. Write the initials of your father or mother, the protein you worked with and the random seed you used for the biased random sequence generator.

(continued on next page)

3. Build a table like

% protein change (input)	% protein identity (output)	% sequence identity from fasta	*e-value*	rank of correct protein	Number of hits with *e-value* < 10^{-5}
0					
30					
50					
70					
80					
85					
90					
100					

The first column is the amount of change you ask your program for.

The second is the sequence identity your program calculates. Do not quote more than 3 significant digits.

The rest of the columns are taken from the fasta output. When you search with the original sequence, you expect it to be at rank 1 in the list of similar proteins. As the sequence changes, it may move down the rank list. Eventually, it may not appear in the rank list.

The numbers in the first column are a suggestion. For small proteins, you may have to use slightly lower numbers to see where homology detection breaks down. For larger proteins, you may still see reasonable alignments with very low sequence identity.

4. If you ask your program to change *x* % of the sequence, you should see that the actual percentage changed is smaller than *x*. (identity > 100 –*x* %) Explain why.

5. What would be a simple change to the structure of the program to make the % change of sequence more controllable ? Describe the change in careful words or pseudocode.