Practice 1: Search for homologous proteins with BLAST

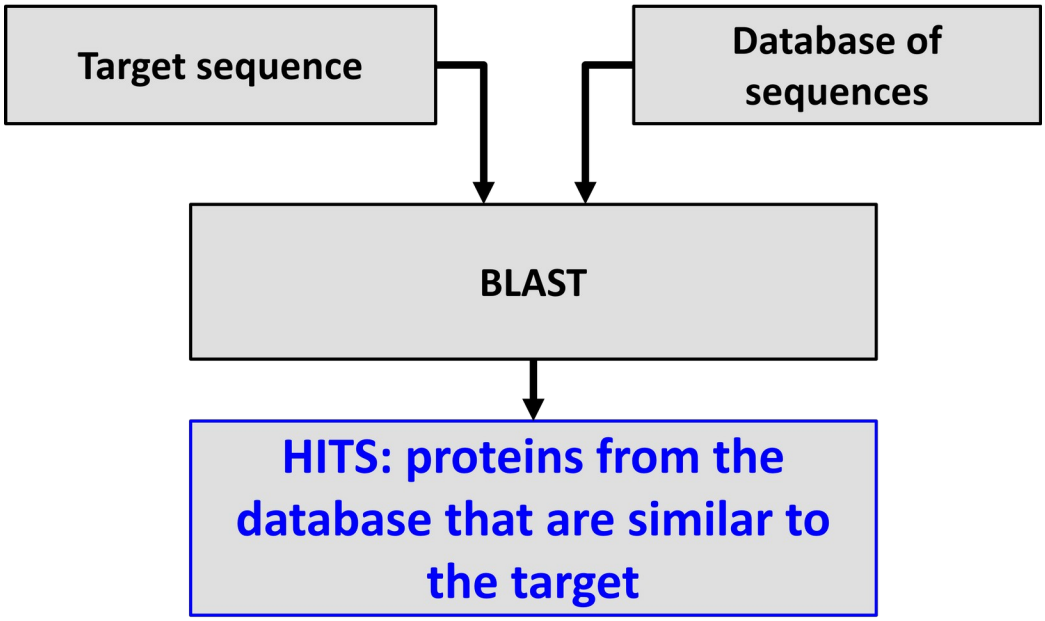
We propose the following problem: We have the sequence of a protein of which we know nothing, neither its function nor structure. This is called protein-problem or target. We want to know if there is a protein with known structure that shares a common ancestor with the target protein (definition of homology). This homologous protein with known structure is called template. We are interested in finding templates because we can use them to model the structure of the target protein. In this exercise we will explore different strategies to find template proteins using BLAST, Hidden Markov Models (HMM) and structure superimposition.

Theoretical concepts:

BLAST (Basic Local Alignment Search Tool): is a program to compare biological sequences and perform local sequence alignments. We will use it to find similar sequences to our target sequence in databases of sequences. BLAST works following these steps:

1. Splits the target sequence into smaller subsequences, also known as words.
2. Matches these words on the database sequence according to a score.
3. The alignments between the words and the database sequences are extended. On each extension step the score of the alignment changes.
4. If the alignment score decreases under a certain threshold the extension stops.
5. All the alignments that show a statistically significant score are returned as hits.

All the protein sequences identified as similar to the target protein are called **hits**.



BLAST provides two statistical measurements for each one of these hits: the score and the expected value (or e-value). High scores will indicate that the two proteins have similar sequences. The e-value is computed as a function of the score and it is the probability of finding a hit with that score just by chance. Besides the score, the computation of the e-value takes into account other parameters such as the size of the database or the length of the alignment.

The e-value can be expressed by the next formula, where n and m are the number of residues of the sequences being compared, N is the size of the database, S is the score, μ is the average of scores and K and λ are other parameters.

$$E(S) \approx \exp\left(-Nmne^{-(S-\mu)}\right)$$

BLAST computes scores by using substitution matrices. Substitution matrices assign a score to each one of the aligned residues in an alignment. In this tutorial we will use BLAST and PSI-BLAST. PSI-BLAST is the iterative version of BLAST and obtains more specific substitution matrices leading to a better performance finding homologous proteins.

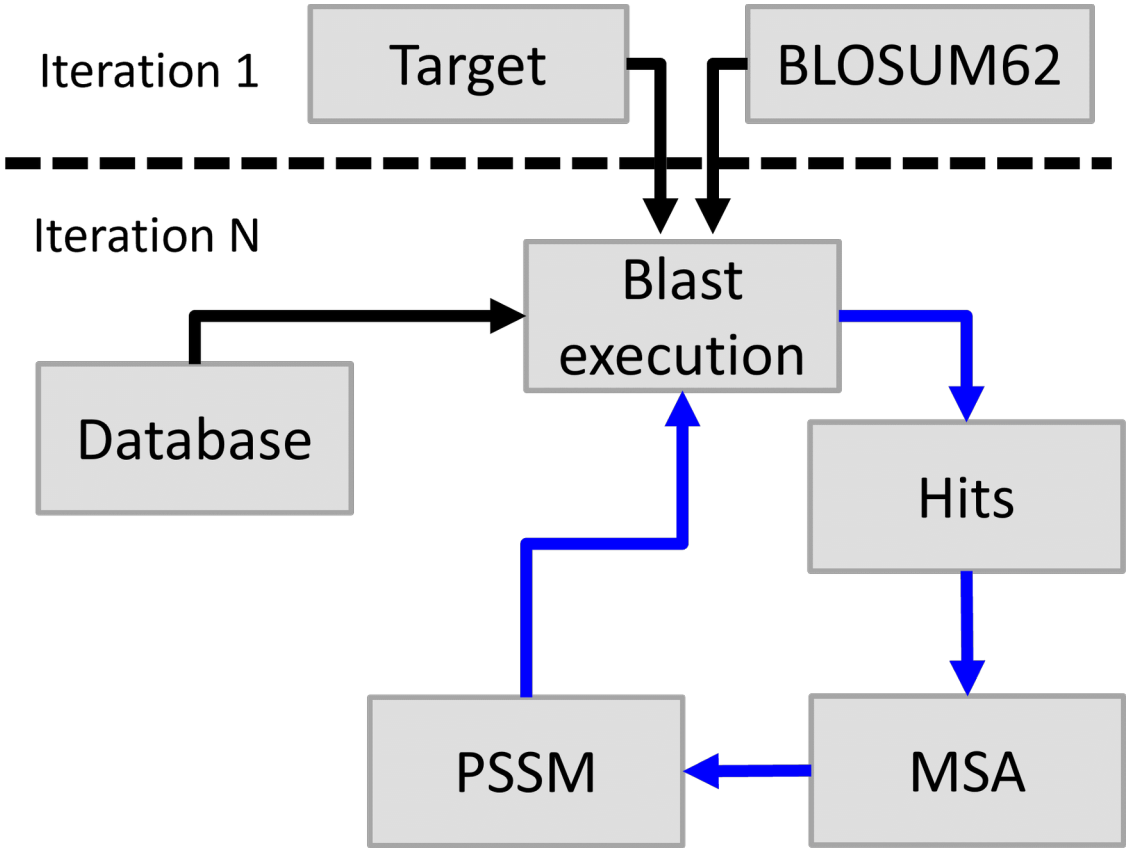
Substitution matrix: Substitution matrices contain scores associated to the frequency of substitution between two amino acids. For example, if you have glutamate (E) aligned with aspartate (D) the score of the aligned pair is M (E, D), where M comes from the substitution matrix.

Substitution matrices are obtained from multiple sequence alignments (MSA). They contain the information regarding residue substitution and conservation of the MSA. According to the information that contain, substitution matrices can be:

- **Non position-specific:** substitution scores are the same for all the positions in the alignment. To obtain non position-specific substitution matrices you compute the substitution frequencies for all positions in the MSA at once.
- **Position-specific:** substitution scores are different for each one of the positions of the alignment. It is like having a different non position-specific substitution matrix for each one of the positions in the alignment. To obtain position-specific substitution matrices you compute the substitution frequencies for each one of the positions in the MSA separately.

One of the most widely used matrices is Blosum62. This has been obtained by aligning blocks of similar sequences, presumably indicating the evolutionary changes of the residues of a protein. Blosum62 is a non position-specific substitution matrix and is the default substitution matrix used by BLAST.

During this tutorial we will use PSI-BLAST, the iterative version of BLAST. PSI-BLAST starts to search for homologous proteins with a BLOSUM62 matrix. Then, finds the hits, aligns their sequences in a MSA, and from this alignment obtains a position specific substitution matrix (PSSM). Afterwards, PSI-BLAST will search again for homologous sequences but using the PSSM computed previously in stead of the BLOSUM62 substitution matrix. PSI-BLAST works by searching new hits and updating its substitution matrix at each iteration. This enables that, after several iterations, the PSSM is specific for the type of substitutions that happen in the homologous proteins to our target. Therefore, hits are found with higher specificity. The next scheme shows how is PSI-BLAST performing:



Tutorial:

Step 1: Using BLAST

Within "exercise_2" you will find the subdirectory BLAST. Within this there is the sequence problem named "target.fa".

To look for proteins of known structure similar to the target protein, try:

```
blastp -query target.fa -db /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq
-out target_pdb.out
```

Example of BLAST usage:
➢ **blastp -query [target_fasta_format]-db [database]-out [output]**

You can see the result of the search in the output file target_pdb.out.

Step 2: Using PSI-BLAST

So far, we have been only using Blosum62. This substitution matrix is generic; therefore we always use the same matrix for all our searches. How could we improve the specificity of our sequence search?

> Using a matrix that is only informative for the substitutions in the protein family of our target sequence.
>
> Using a matrix that contains specific substitution scores for each one of the positions of the alignment.

These improvements are achieved using a **Position Specific Substitution Matrix** or PSSM. But, how can we obtain this matrix?

PSSMs are obtained from **Multiple Sequence Alignments** (or MSA). Within the MSA, the probabilities and scores for each possible substitution for all the positions of the MSA are computed. This MSA must contain the target sequence to indicate the reference position of the residues within the MSA.

One way to automatically perform a sequence search with PSSM is to use **PSI-BLAST**. This program is an **iterative** version of BLAST. It looks for sequences similar to the target sequence, it aligns them into a MSA and with this it recalculates a new substitution matrix. The matrix is stored and used to add more similar sequences.

Iteration 0 → Target + Blosum62 → [similar sequences] 0 → MSA 0 + Target → PSSM 0

Iteration 1 → PSSM 0 → [similar sequences] 1 → MSA 1 + Target → PSSM 1

Iteration 2 → PSSM 1 → [similar sequences] 2 → MSA 2 + Target → PSSM 2

.....

Example of PSI-BLAST usage:

➢ **psiblast -query [target_fasta_format] -db [database] –num_iterations [number of iterations] -out [output]**

Using the previous example, we can make an iterative search into the PDB database. We are going to make 5 iterations:

```
psiblast -query target.fa -db /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq
-num_iterations 5 –out target_pdb_5.out
```

Results per iteration are indicated by the word “Round”. You can search this word in the output file to compare the results at different iteration rounds.

But this raises a problem: the PDB database contains only sequences with known structure. This implies that is a very **redundant database**. Most of the PDB proteins belong to a small number of species (human and mouse, among others) and in many cases you find the same protein repeated several times. Besides, some protein families have been crystallized several times and have many structures, while many others don’t.

Then, if you obtain a PSSM using this **biased** set of sequences, your sequence search will be biased too. This PSSM will shift your results towards those families with higher number of known structures instead of incorporating the evolutionary information about substitutions, deletions and/or insertions.

To solve this problem we need a **non-redundant database of protein sequences**. SwissProt or Uniprot are examples of non-redundant protein sequence databases. They contain a wide spectrum of protein sequences for a lot of different species. Also, they are much bigger than the PDB. The PSSMs built using these databases will represent the evolutionary information of the target protein family with increased accuracy, and this will improve the results of our sequence search.

So, to perform an unbiased sequence search we will carry out the following strategy:

Run PSI-BLAST on the SwissProt database and store the PSSM generated on the 5th iteration. This can be done with the **–out_pssm** option. Do:

```
psiblast -query target.fa -num_iterations 5
-out_pssm target_sprot5.pssm -out target_sprot_5.out
-db /mnt/NFS_UPF/soft/databases/blastdat/uniprot_sprot.fasta
```

Apply the obtained PSSB to make a sequence search in the PDB database. The obtained results will be proteins with known structure. This can be done with the option **–in_pssm**. Do:

```
psiblast -db /mnt/NFS_UPF/soft/databases/blastdat/pdb_seq -in_pssm
target_sprot5.pssm -out target_pdb_sprot5.out
```

Now we don't make iterations in the PDB database. This is because if we iterate a new PSSM matrix, this would be built with the PDB sequences and the evolutionary significance would be biased again.

Note that in the second execution of PSI-BLAST we are not including the query into the command. This is because the program is including automatically the information of the target sequence in the PSSM used to perform the search.

Other options of "psiblast" are:

- evalue: E-value threshold to see output sequences (default 10)
- inclusion_ethresh: E-value threshold to include sequences in the PSSM (default 0.002)

To see all options by doing "psiblast -help"

Step 3: Using PSI-BLAST with alighnments of ClustalW

We know that PSSM are obtained from MSA. This MSA can be obtained by PSI-BLAST or by a sequence alignment program, such as **ClustalW**. Using ClustalW enables you to specifically select the sequences whose alignment will build your PSSM. Then, you can introduce the MSA generated by ClustalW into PSI-BLAST and run a sequence search on PDB.

Using ClustalW is extremely simple. Just concatenate the sequences you want to align in fasta format into a file and then type clustalw and the name of the file, like this:

➢ **clustalw file.fa**

Now we will get the fasta sequences for a set of selected PSI-BLAST outputs. For that we will use the program **FetchFasta.pl**. This program takes as input a list file that **YOU HAVE TO GENERATE** by copying and pasting several lines of an output from a PSI-BLAST search on SwissProt (target_sprot_5.out in the tutorial). You can use "vi" or "nano", and it may be saved with a name such as file.list. For example,

the format of the file “file.list” with sequences from the database “target_sprot_5.out ” looks similar to this:

```
ORC1_DROME (O16810) ORIGIN RECOGNITION COMPLEX SUBUNIT 1 (DMORC1).    380   e-105
ORC1_SCHPO (P54789) ORIGIN RECOGNITION COMPLEX SUBUNIT 1.             295   2e-79
ORC1_CANAL (O74270) ORIGIN RECOGNITION COMPLEX SUBUNIT 1.             223   1e-57
ORC1_YEAST (P54784) ORIGIN RECOGNITION COMPLEX SUBUNIT 1 (ORIGIN...   189   1e-47
ORC1_KLULA (P54788) ORIGIN RECOGNITION COMPLEX SUBUNIT 1.             179   1e-44
CC18_SCHPO (P41411) CELL DIVISION CONTROL PROTEIN 18.                 115   2e-25
CC6_YEAST (P09119) CELL DIVISION CONTROL PROTEIN 6.                    87   8e-17
YPZ1_METTF (P29570) HYPOTHETICAL 40.6 KDA PROTEIN (ORF1').             60   1e-08
YPV1_METTF (P29569) HYPOTHETICAL 40.7 KDA PROTEIN (ORF1).              60   1e-08
SIR3_YEAST (P06701) REGULATORY PROTEIN SIR3 (SILENT INFORMATION ...    47   1e-04
G6PI_OENME (P54243) GLUCOSE-6-PHOSPHATE ISOMERASE, CYTOSOLIC (GP...    31   6.6
```

Then use the following command:

```
perl /mnt/NFS_UPF/soft/perl-lib/FetchFasta.pl -i file.list

-d /mnt/NFS_UPF/soft/databases/blastdat/uniprot_sprot.fasta -o file.fasta
```

Now, we need to obtain the sequence alignment of these sequences plus the target sequence. First we have to concatenate all fasta sequence in one file. **DO NOT FORGET TO INCLUDE THE TARGET SEQUENCE**. The target sequence has to be included in the file because is required to indicate the position of the residues in the PSSM. It is important that the first sequence in the file is the target sequence.

This can be done by running:

```
cat target.fa > pssm.fasta
```

```
cat file.fasta >> pssm.fasta
```

Then run clustalw:

```
clustalw2 pssm.fasta
```

Then, change the format of the alignment to fasta format:

```
perl /mnt/NFS_UPF/soft/perl-lib/aconvertMod2.pl -in c -out f
<pssm.aln>pssm.fa
```

Then, we introduce this MSA as input to PSI-BLAST using the option **-in_msa** with this file:

```
psiblast -in_msa pssm.fa -out target_pdb_specific.out -db

/mnt/NFS_UPF/soft/databases/blastdat/pdb_seq
```

QUESTIONS FROM THE TUTORIAL

Now we can compare all the results and answer the following questions:

1) Why are the e-values different in *target_pdb.out* than in the fifth iteration in *target_pdb_5.out*?

2) Why do we need to run psiblast with uniprot_sprot.fasta before searching in pdb_seq?

3) When obtaining the file *target_pdb_sprot5.out* why we didn't run 5 iterations as before?

4) Search in the SCOP database with the PDB code of the best match of the target sequence. Do all the files *target_pdb_specific.out, target_pdb_sprot5.out, target_pdb_5.out* and *target_pdb.out* produce the same result?

5) Can you use the file target_sprot5.out to obtain the name of the fold in SCOP? Why?

6) What are the folds of the following sequences?

    a. *problem1/serc_myctu.fa*

    b. *problem2/p72_mycmy.fa*

    c. *problem3/lip_staau.fa*

    d. *problem4/orc1_human.fa*