Practical 'Fundamentals of Bioinformatics':

In this project you will be made aware of common problems in Bioinformatics by first-hand experience. Some of it may be hard for you to understand because you don't know the biological background. Some of it may be difficult for you to do because you don't know how to program it. That is the point of this exercise; Bioinformatics is at the interface of biology and computer science. You're not expected to know all of that, yet. But you are expected to find out how far your current knowledge reaches. That is where you will start learning!

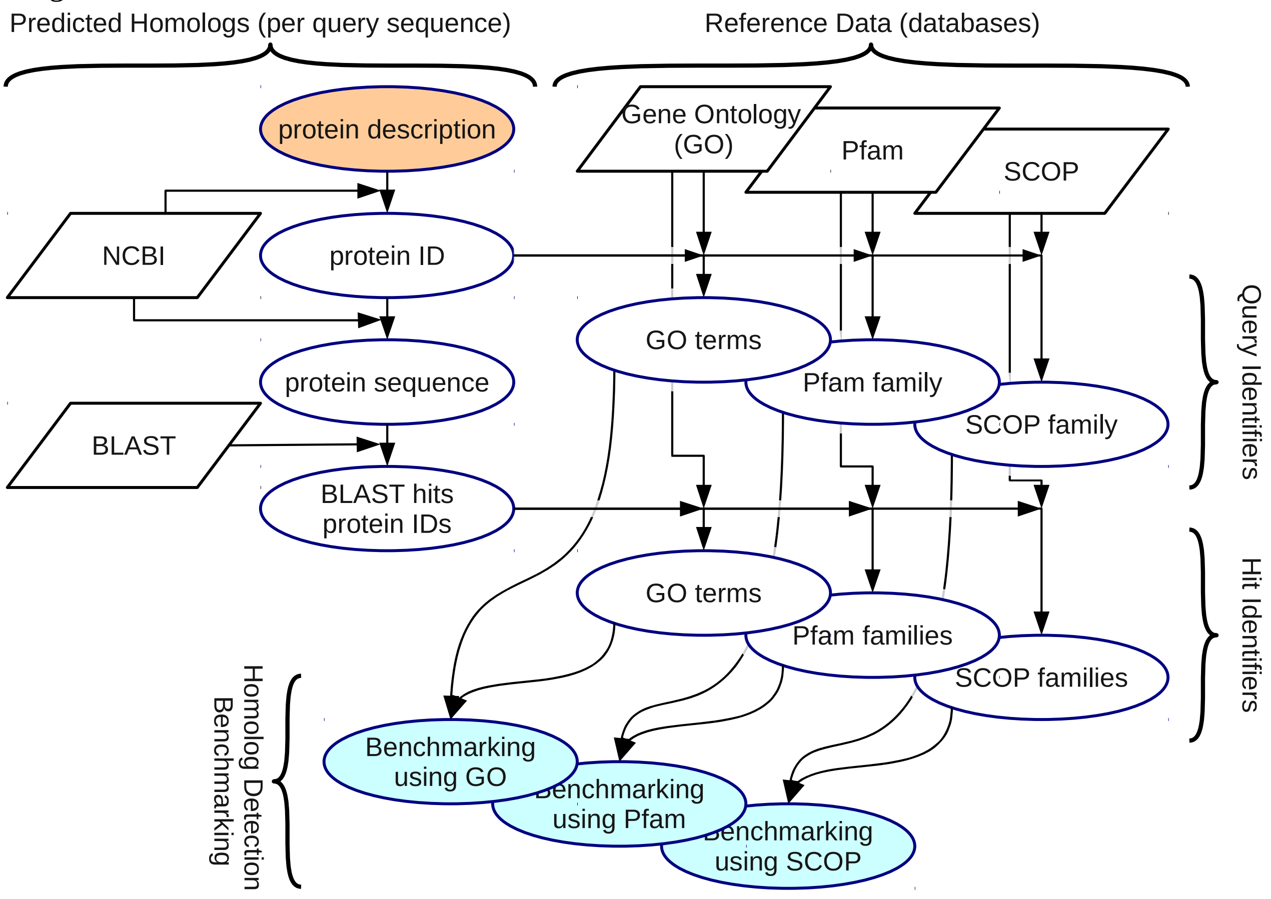
Please also be aware that, in general, there is much more to be said about the tools and analysis techniques you are going to use in this Practical. In part, that is what the practical is about; not only to find the limits of your own knowledge, but also the limitations of available data and knowledge in general. For a large part, subsequent courses will delve much deeper into the details of the tools and data used here.

Main theme: “Homology detection in proteins”

Problem case: “Benchmarking BLAST homology searches”

Workflow:

At a glance:



Below you will find the workflow in much more detail. Technical details and further background are described in the '**Technical Manual**', be sure to us that as a *reference*. In addition, it contains several *tutorials* which you should work through. These will give you some first-hand experience with what kinds of data the databases contain, how to access them and what results you can expect.

1) **Find your protein (Practical FoB 2)**

Starting from the description of possible protein families, you find the matching protein IDs from SwissProt/UniProt that uniquely identify a particular protein. Find the proteins which gene names match:

- "EGR-1" from M. musculus
- "Calmodulin" from H. sapiens
- "OmpF" from E. coli (transmembrane protein)
- "'secD" from E. coli (transmembrane protein)
- "Sensory Rhodopsin 1" from *H. salinarum*. (transmembrane protein)

Check that you have at least one protein from the 'α' structural class and at least one from the 'β' class. From both classes (α and β) you should have at least one protein with a high-resolution structure.

Use the NCBI server (www.ncbi.nlm.nih.gov) to search using the criteria above. Retrieve your protein (UniProt) IDs and sequences, these will be your query proteins. Make sure you store all your subsequent results such that you can link them back to the query proteins (e.g. query protein in its own directory). As an alternative to NCBI you can use the Sequence Retrieval System (SRS) at http://srs.ebi.ac.uk/**.**

2) **Find GO terms for your proteins (Practical FoB 2)**

Write a python script to retrieve the GO terms from the Gene Ontology (GO) database www.geneontology.org for each query protein. Easiest programmatic access is through QuickGO.

Questions:

1. How are terms organized in the Gene Ontology database?
2. Can you have multiple GO terms and sections for one protein? Why and how (not)?
3. Explain the biological rationale behind the GO database.
4. Which GO terms and section(s) did you retrieve for each of your query proteins? Is there a set of Go terms that uniquely identifies the sequence family of your protein? Which ones?

3) **Find the Pfam family for your proteins (Practical FoB 2)**

Write a python script to retrieve the Pfam family ID(s) and family info from the Pfam database http://pfam.sanger.ac.uk/ for each query protein.

Questions:

1. How are proteins organized in the Pfam database?
2. Can you have multiple Pfam families for one protein? Why and how (not)?
3. Explain the biological rationale behind the Pfam database.
4. Which Pfam families and clans did you retrieve for each of your query proteins?

4) **Find the SCOP family for your proteins (Practical FoB 4 – n.b. later!)**

Write a python script to retrieve the SCOP family and info from the SCOP database http://scop.mrc-lmb.cam.ac.uk/scop/ for each query protein.

Questions:

1. How are proteins organized in the SCOP database?
2. Can you have multiple SCOP classifications for one protein? Why and how (not)?
3. Explain the biological rationale behind the SCOP database.
4. Which SCOP classification(s) did you retrieve for each of your query proteins?

5) **Find matching sequences (Practical FoB 3)**

Write a python script to retrieve matching sequences from the SwissProt/UniProt database using NCBI BLAST http://blast.ncbi.nlm.nih.gov/ for each query protein. Use the script developed in the parsing assignment on the first day to further process the BLAST results returned from your query; particularly you will need UniProt IDs for each hit. You will also want to conserve ordering of the hits (see below at 7).

Questions:

1. Explain what is meant by homology.
2. Explain what the purpose of BLAST is in the context of homology. How can you relate this to the benchmarking exercise? [Perhaps not here]
3. What is a BLAST e-value?
4. How are proteins organized in the SwissProt/UniProt database?
5. Can you have multiple UniProt IDs for one protein? Why and how (not)?
6. Explain the biological rationale behind the SwissProt/UniProt database.
7. Give a brief summary of the hits retrieved for each of your query proteins.

6) Find GO terms and Pfam (and SCOP) families for each matching sequence (Practical FoB 3)

Use the scripts developed at (2), (3) and (4) to retrieve GO terms and Pfam and SCOP families for all hits obtained in (5) for each of your query proteins.

7) Score (benchmark) effectivity of sequence matching to identify (real) homologs (Practical FoB 5)

All matching sequences for a query protein are your predicted (putative) homologs. You consider each of the three databases (GO, Pfam and SCOP) as authoritative sources for homology information. This allows you to score the efficacy of homology detection by sequence comparison (this means: BLAST) against other (expert) information. Your (BLAST) predictions divide all proteins into two classes: homologous to the query, or not. Your database (GO, Pfam or SCOP) divides all proteins into three classes: a true homolog (if query and BLAST hit are in the same category in the database), not a true homolog (in different categories), or unknown (the BLAST hit is not in the database!). This is easier to follow in a table:

	database entry:	*hit in same category as query*	*hit in different category as query*	*hit not in database*
data:	**conclusion:**	true homolog	no homolog	unknown
BLAST hit	*putative homolog*	TRUE POSITIVE (TP)	FALSE POSITIVE (FP)	UNKNOWN
no hit	*putative non-homolog*	FALSE NEGATIVE (FN)	TRUE NEGATIVE (TN)	UNKNOWN

When comparing predictions with a 'gold-standard' truth, there are two measures that are most important (at least, they are commonly used): *Coverage* and *Error*. Coverage means, how much of the known things did you predict (or, how many did you miss). Coverage is also known as *Sensitivity*. Error means, how many of your predictions are wrong (i.e., the predictions do not match the known things). In stead of Error, also sometimes *Precision* is used as Precision = 1 – Error. Precision is also known as *Specificity*. Using the table above, you can work out that:

Coverage = TP/(TP+FN)

Error = FP/(FP+TN)

You should work this through and verify the formula's. Then, you can work out how, using the list of IDs from the BLAST hits, and the category information from the database, you can count TP, FN,

FP and TN.
The final thing that we will be using is the hit score (BLAST e-values). The list you retrieve from the BLAST server should already be ordered by this score (but it does not hurt to check that!). The first hit should be the most reliable prediction. At each item in the hit list, you count how many of the preceding hits were true (and thus are TP; predicted and right) and how many false (thus FP; predicted but wrong). This TP *vs.* FP datapoints as a function of hit score. With FN and TN (which you have to figure out how to get) you can calculate Coverage and Error. Plotting these against each other gives you a so-called Receiver Operator Characteristics (ROC) plot. See http://wikipedia.org/wiki/Receiver_operating_characteristic for a thorough explanation.

Questions:

1. Describe for each of the databases (GO, Pfam, SCOP) you compare against, how you define your Positives and Negatives. Also describe how, from these, you define TP, FP, TN and FN.Discuss some of the different assumptions behind the GO, Pfam and SCOP databases, and how these assumptions are relevant for benchmarking homology detection. Include at least a discussion on over- and under-prediction.
2. Discuss the suitability of GO, Pfam and SCOP as a reference to benchmark BLAST for determining homolog relationships between proteins. You should think about, the biological relevance, reliability, the number of reference sequences in each database.
3. In Pfam there is the family and clan level. In GO can be made at different levels in the term hierarchy. SCOP classifies at family, super-family, fold and class. What does it mean for your definition of a true homolog, if you start comparing proteins at these different levels? You can draw some general conclusions, but also about the different (homology) definitions on which the databases are constructed.

(Practical FoB 6)
You will be given some scientific literature related to transmembrane proteins & homology. The papers need to be discussed in your report.

Deliverables:

During this practical project you will be creating several (python) scripts. Also, you should record your work in a concise report. In the report you should also answer the *questions* above. The scripts and your report together must be handed in at the end of the course, and will count towards your final mark. As an appendix, you must also supply the IDs of the sequences retrieved and all annotations (GO, Pfam, SCOP), as well as the data for the ROC plots. Your code must be well readable, which means structure, explanatory variable and function names and comments. File names and query entries may not be hard-coded.