


Project Tutorial

Benchmarking BLAST and PSI-BLAST homology searches

Introduction

This project will give you first-hand experience in solving common problems in Bioinformatics. Bioinformatics is an interdisciplinary field combining biology and computer science. Depending on your background some parts of this project may be unfamiliar and challenging for you. The objective of this project is for you to learn these parts and to see how far your current knowledge reaches. However, this project is not only about finding 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.

Time plan

Week 1	Mon 4 Sept	Introduction to project + make groups
	Tues 5 Sept	Getting started on the VM + Step 1. Set up a local sequence database
Week 2	Mon 11 Sept	Step 2. Find putative homologs using (PSI-)BLAST
	Tue 12 Sept	Cont. step 2 + choose database (GO, Pfam or SCOP) + start writing
Week 3	Mon 18 Sept	Step 3. Create a gold standard based on GO, Pfam or SCOP
	Tue 19 Sept	Step 4. Benchmark (PSI-)BLAST to your gold standard 
Week 4	Mon 25 Sept	Project work
	Tue 26 Sept	Project work and <u>hand in draft report</u>
Week 5	Mon 2 Oct	Project work and <u>hand in feedback of draft report</u>
	Tue 3 Oct	Project work
Week 6	Mon 9 Oct	Project work and <u>hand in final report</u> + make discussion groups
	Tue 10 Oct	Discussion session
Week 7	Mon 16 Oct	Discussion session
Week 8	Fri 27 Oct	<u>Final presentations</u>

Deliverables

This project counts as 30% of the final course grade. The deliverables that are listed below all have to be handed in and are either graded or pass/fail. You will be assigned to give feedback on the draft report of another group and your group's draft report will get feedback from members from other groups. On the final day of the course, you will present the project work together with your group. Below, each of these deliverables is explained in greater detail.

- Draft report (pass/fail)
- Feedback of draft report (pass/fail)
- Final report (2/3 of final project grade)
 - Length (pass/fail)
 - Questions (pass/fail)
 - Scripts (pass/fail)
- Presentation (1/3 of final project grade)

Draft report (hand in in PDF format via Canvas)

The draft report must contain between 1000 and 1500 words and contain following sections: *introduction*, *materials & methods* and (preliminary) *results & discussion*. The *results and discussion* should include a ROC plot and its interpretation. You must clearly state your research question in the *introduction* and answer it in the *results & discussion*.

Review draft report of another group (hand in in PDF format via Canvas)

You will be assigned to give feedback on the draft report of another group. The criteria for this will be provided to you later. Likewise your group's report will get feedback from members from other groups, which you can use to improve your final report.

Final report (hand in in PDF format via Canvas)

The final report must contain the following sections:

- Abstract (max 200 words):
 - Motivation, results & impact
- Introduction:
 - Include references to previous studies from literature
 - Explain the background of (PSI-)BLAST and your database
 - State your research question
- Materials & Methods:
 - Include a flow chart
- Results & Discussion:
 - Include your ROC plot and its interpretation.
 - Answer your research question
 - Compare your results to previous results from literature
 - Discuss the (dis)advantages of using your chosen database as a gold standard for benchmarking homologs found with (PSI-)BLAST compared to the other two databases.

- Conclusions
 - Main conclusions & future research
- Tables & figures
 - Explain all axes, labels, lines points, in the caption of your figure/table.
 - Refer to each figure/table in the main text, and explain in the main text what can be seen from the figure/table.

The final report must contain between 1500 and 2500 words. We count everything (including figure text) except references.

All the questions from this manual should be answered in the suggested sections in your final report. Please indicate in brackets (e.g. [Q1]) where in the text you answer each question.

All scripts you produce or complete during the practical should be handed in alongside the final report. Your code must be readable, which means it should be well structured, use self-explanatory variable and function names, and be sufficiently commented. File names, paths and query entries may not be hard-coded.

The final report must clearly state what each group member contributed to the project. Individual students grades may be adjusted according to the reported and observed differences in workload.

The final report is graded based on: abstract, context, contents, research questions, literature, structure and layout.

Presentation

In the presentation you should compare your results to the results of the other groups (*Hint: Show ROC plots!*). To this end, all reports will be made available on Canvas. You should make a PowerPoint presentation containing at most 7 slides, to support discussion of the following subjects:

- Benchmarking with your database.
- Compare your results with the groups that worked on the same database as you.
- Explain the reason for differences in the results.
- Compare the different results from all three databases. What are the advantages and disadvantages of using your chosen database as a gold standard? (*Hint: See Q4.4*)
- How well do you think the benchmarking answers the research question?
- Future directions?

The presentation should last 10 minutes, followed by some time for discussion. *Every student* is expected to ask at least two questions to a group using a different database than their own.

Step 1. Set up a local sequence database

Create directories

Before we can start performing (PSI-)BLAST searches locally, we have to create directories to store our query sequences, databases, and search results. You can again create these directories from the command line.

```
$ mkdir queries
```

```
$ mkdir db
```

```
$ mkdir results
```

Download the 209 protein sequences

We have provided you with a file of 209 UniProt IDs and a skeleton script (`fetch_sequences.py`) to download the sequences in FASTA format of all the proteins in the list of UniProt IDs. The script should store these sequences into a single FASTA file in the *db* directory you have just created. Simultaneously, it should store the sequences as separate FASTA files in the *queries* directory. Complete the script and execute it.

Format local sequence database

One of the main advantage of running the standalone version of BLAST is that you can create your own sequence database. You can build your database out of any FASTA formatted file containing multiple proteins: the FASTA file containing all 209 sequences in our case. Use the “makeblastdb” program to format your local sequence database. Review the options available for this program by typing the following command:

```
$ makeblastdb -h
```

[Q1] It is important to know how big your database is. Invoke *makeblastdb* with the *-logfile* option to confirm the number of sequences correctly formatted. Which format do the newly generated files have? What information do they contain? [Materials and Methods]

Step 2. Find putative homologs using (PSI-)BLAST

Explore blast2 and blastpgp searches on P00698

The next step is to search for putative homologs of your proteins using blast2 (standard BLAST) and blastpgp (PSI-BLAST). To find out about all options that are available for these programs, type the following commands in your command-line interface:

```
$ blastp -h
```

```
$ psiblast -h
```

We will explore what these options do using protein P00698 as a test case. Perform a default BLAST search of P00698, using the following command:

```
$ blastp -query queries/P00698.fasta -db db/<name_of_your_db>
```

Now run the PSI-BLAST using the following commands and compare the results.

```
$ psiblast -query queries/P00698.fasta -db db/<name_of_your_db>  
-num_iterations 3
```

To produce a more concise overview of the output of either blast2 or blastpgp, add the following option:

```
-outfmt 6
```

Note that in order to run PSI-BLAST, the -j parameter must be set to something greater than 1. The default of -j 1 means that there are no iterations and that it is therefore the same as a single BLAST search.

Automate blast2 and blastpgp searches on all you queries

Now that you have familiarized yourself with invoking (PSI-)BLAST queries from the commandline, complete the provided skeleton script (run_local_blast.py) to automate the process for every file in your queries directory. The output should be written as tab-separated UniProt IDs and BLAST scores (e-values), with NA if no match was found between proteins given a certain e-value cut-off. Example:

P000001	P000002	0.0001
P000001	P000003	10.0
P000002	P000003	NA

[Q2.1] Explain what is meant by a homolog and what the purpose of BLAST is in the context of homology! *[Introduction]*

[Q2.2] Explain the main differences between BLAST and PSI-BLAST! *[Introduction]*

[Q2.3] What is the meaning of the bit-score and the corresponding e-value? Explain how the e-value is calculated! *[Materials & Methods]*

[Q2.4] Is it possible that PSI-BLAST stops before it reached the number of iterations specified by the -j parameter? If so, why? *(Hint: Consider the purpose of the -h parameter.) [Materials & Methods]*

[Q2.5] What is the difference between e-value threshold parameters -evalue and -inclusion_ethresh in psiblast? *[Materials & Methods]*

[Q2.6] Try different e-value thresholds, and explain your observation! *[Results and Discussion]*

[Q2.7] Use the e-value distribution that your script produces to explain the main differences between BLAST and PSI-BLAST! *(Hint: With which method do you expect more hits for the same query sequence? Which method is more sensitive? Which method is more specific?) [Results and Discussion]*

[Q2.8] Suppose you were not using a local sequence database for your PSI-BLAST runs. Which database would be best to generate a PSI-BLAST profile (PSSM) with, and what search strategy would you use? *[Introduction or Results and Discussion]*

Step 3. Create a gold standard

Biologists commonly use sequence alignment programs like BLAST to infer whether two proteins are functionally similar. Typically, a sequence identity larger than 35% is considered to be a strong indication for functional similarity, whereas a sequence identity between 20% and 35% correspond to the so called twilight zone, in which case there is not enough evidence to infer functional similarity or dissimilarity.

There are however alternative ways to predict if proteins share functional similarity. In the following step, you will retrieve expert information about your proteins from one of the three databases. This information will be used as a gold standard in *step 4*, where you score (benchmark) the accuracy of your (PSI-)BLAST results.

Your group now has to choose one of the following databases, from which you will retrieve such information about your proteins: Gene Ontology (GO), Pfam or SCOP. Depending on which database you choose to work on, continue with one of the following steps: 3a, 3b or 3c.

Step 3a. Using the Gene Ontology (GO) database as gold standard

The Gene Ontology database is a community effort to establish a standardized representation of gene and gene product attributes across different species. As the name suggests, GO is an ontology. An *ontology* is an explicit specification of how terms in a specific domain are organized. In GO the domain consists of terms annotating genes and gene products. These terms are organized in a specific way, which is explained in greater detail on GO's website: www.geneontology.org

An alternative way to infer functional similarity is by comparing GO terms. For example, LAP3 (UniProt ID: Q01532) and BLMH (UniProt ID: Q13867) are functionally related. The idea is that the more GO terms two proteins share, the more likely it is that they have similar functions. We can design a score taking this aspect into account. Preferably the score should be normalized, e.g. between 0 and 1. Given a pair (p_1, p_2) a possible scoring function is:

$$s(p_1, p_2) = \frac{g(p_1) \cap g(p_2)}{g(p_1) \cup g(p_2)}$$

where $g(p)$ is the set of GO terms associated to protein p .

[Q3a.1] How are terms in the Gene Ontology database organized?
[Introduction]

[Q3a.2] Can you have multiple GO terms per protein? Why (not)? (Hint: Consider the case that two proteins share exactly the same function, but have different sequences.) [Introduction or Materials & Methods]

[Q3a.3] Explain the biological rationale behind the GO database!
[Introduction]

[Q3a.4] Explain in detail how it is decided what *true positives* are based on this gold standard database! Also give an example of some problematic protein assignments! (Hint: Why is it in general, convenient that scores are normalized? What would be the score of a protein when compared to itself?)
[Materials & Methods]

GO gold standard

Consider the all-against-all (PSI-)BLAST searches you did before. Instead of using BLAST to determine if proteins are functionally similar, we now want to do so using the protein's GO annotations. Complete and execute the skeleton script (classify_go.py) to automate this process for the remaining protein pairs. Determine suitable score thresholds by plotting a distribution of your calculated scores. Indicate the thresholds you chose in this plot, and add it to your report's *Results* section.

Step 3b. Using the Pfam database as gold standard

Pfam classifies proteins into families and clans, based on the motifs, repeats and domains they contain. If two proteins are indeed homologous, one would expect them to have the same motifs, repeats and domains, and thus be assigned to the same families. We will use these families to which Pfam assigns proteins to assert if they are indeed homologs.

The URL access to Pfam by UniProt ID is very simple. P00698 for example, would be accessed via <http://pfam.xfam.org/protein/P00698?output=xml>

The reply will look something like:

```
<!--
  information on UniProt entry P00698 (LYSC_CHICK), generated: 11:42:03 01-Jul-2015
-->
<pfam xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://pfam.xfam.org/"
xsi:schemaLocation="http://pfam.xfam.org/
http://pfam.xfam.org/static/documents/schemas/protein.xsd" release="28.0"
release_date="2015-05-15">
  <entry entry_type="sequence" db="uniprot" db_release="2014_07" accession="P00698"
  id="LYSC_CHICK">
    <description>
      <![CDATA[ Lysozyme C EC=3.2.1.17 ]]>
    </description>
    <taxonomy tax_id="9031" species_name="Gallus gallus (Chicken)">
      Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Testudines +
      Archosauria group; Archosauria; Dinosauria; Saurischia; Theropoda; Coelurosauria; Aves;
      Neognathae; Galliformes; Phasianidae; Phasianinae; Gallus.
    </taxonomy>
    <sequence length="147" md5="e485279d5bf21caa3b7967c566c68a5a" crc64="81E85743FF579468"
    version="1">
      MRSLLILVLCFLPLAALGKVFGRCELAAMKRHGLDNYRGYS LGNWVCAAKFESNFNTQATNRNTDGSTDY GILQINSRWWCNDGRTP
      GSRNLCNIPCSALLSSDITASVNCARKIVSDGNMNAWVAWRNRCKGTDVQAWIRGCRL
    </sequence>
    <matches>
      <match accession="PF00062" id="Lys" type="Pfam-A">
        <location start="19" end="145" ali_start="19" ali_end="145" hmm_start="1"
        hmm_end="124" evalue="1.2e-49" bitscore="179.40"/>
      </match>
    </matches>
  </entry>
</pfam>
```

Look for lines like this:

```
<match accession="PF00062" id="Lys" type="Pfam-A">
```

accession indicates the Pfam family accession (here PF00062), *id* indicates the name (here Lys), and *type* indicates whether the Pfam annotation is manually curated (Pfam-A) or computationally inferred (Pfam-B). The manual curation is generally more reliable than the computationally inferred. The full description of this interface (part of the 'RESTful' interface) can be found in the Pfam help pages: <http://pfam.xfam.org/help#tabview=tab11>

A nice extension of this part is to also look at the level of Pfam clans: groups of related families. A list of Pfam clans can be found at: <http://pfam.xfam.org/clan/browse>

Pfam gold standard

Write a new python script to retrieve the Pfam family ID(s) for each of your query proteins from the Pfam database. Store the relevant information you retrieve (such as accession numbers, family IDs and type) in an easily accessible format. Finally, decide on a way to score whether or not two proteins are homologs according to information retrieved from Pfam. Your script must create a tab-delimited file similar to `benchmark_dummy_output.txt`, in which you save a list of all possible protein pairs and a calling *similar* or *different*, for the pairs which you respectively do and do not think are homologs according to Pfam.

P000001	P000002	similar
P000001	P000003	different
P000002	P000003	different

[Q3b.1] How are proteins organized in the Pfam database? *[Introduction]*

[Q3b.2] Can you have multiple Pfam families for one protein? Why and how (not)? *(Hint: Domains.) [Introduction or Materials & Methods]*

[Q3b.3] Explain the biological rationale behind the Pfam database! *[Introduction]*

[Q3b.4] Explain in detail how it is decided what *true positives* are based on this gold standard database! Also give an example of some problematic protein assignments! *[Materials and methods]*

Step 3c. Using the SCOP database as gold standard

The structural Classification of Proteins (SCOP) database assigns proteins to families based on properties of their 3D structure. Because form and function are closely related in proteins, such classification can be informative about whether two proteins are functionally similar.

To retrieve SCOP family IDs for a protein, you need PDB identifiers. There are several ways to map UniProt IDs to PDB IDs, but for this practical we have provided you with a lookup table: PDB_ID_lookup.tab. Using the PDB IDs, look up the SCOP classification(s) for your proteins in the parsable flat file version of SCOP (dir.cla.SCOP.txt), found at <http://scop.mrc-lmb.cam.ac.uk/scop/index.html>

Browse the SCOP website for information on how its information is organized. Identify which information is relevant for the purpose of benchmarking (PSI-)BLAST's putative homolog hits. Write a new Python script to map your UniProt IDs to PDB IDs, and parse this information from dir.cla.scop.txt. Store the relevant information you retrieve in an easily accessible format. Finally, decide on a way to score whether or not two proteins are homologs according to information retrieved from SCOP. Your script must create a tab-delimited file similar to benchmark_dummy_output.txt, in which you save a list of all possible protein pairs and a calling *similar* or *different*, for the pairs which you respectively do and do not think are homologs according to SCOP.

[Q3c.1] How are proteins organized in the SCOP database? [Introduction]

[Q3c.2] Can you have multiple SCOP classifications for one protein? Why and how (not)? (*Hint: Domains.*) [Introduction or Materials & Methods]

[Q3c.3] Explain the biological rationale behind the SCOP database! [Introduction]

[Q3c.4] Explain in detail how it is decided what *true positives* are based on this gold standard database! Also give an example of some problematic protein assignments! [Materials & Methods]

Step 4. Benchmark (PSI-)BLAST to your gold standard

Every matching sequence with an e-value below a chosen threshold, or hit, that (PSI-)BLAST finds for your queries are that query's predicted (putative) homologs. This divides your proteins into two categories: they are putative homologs to your query according to (PSI-)BLAST, or they are not. In this last step, you will use the expert information you've retrieved and interpreted in step 3 as a gold standard for homology prediction. Your interpretation of the GO, Pfam or SCOP data divides every pair of proteins into three categories: they are true homologs, they are not true homologs, or you can't say given the information from the database (e.g. because your interpretation is ambiguous, or because one of the proteins wasn't found in the database). We can then score each (non-)homolog predicted by (PSI-)BLAST as a True Positive (TP), False Positive (FP), False Negative (FN) or a True Negative (TN), by comparing it to the conclusion drawn from your gold standard.

We will define a protein pair that is homologous as a positive, and a protein pair that's different as a negative. If a protein pair is correctly predicted to be homologous by (PSI-)BLAST, it is called a true positive. If (PSI-)BLAST predicts a protein pair as homologous, while your gold standard says they are not, it is called a false positive. This is all summarized in the table below:

	Database entry:	Hit in the same category as query	Hit in different category as query	Hit not in database
Data:	Conclusion:	True homolog	Non-homolog	Unknown
(PSI-)BLAST hit	Putative homolog	TRUE POSITIVE	FALSE POSITIVE	UNKNOWN
No hit	Putative non-homolog	FALSE NEGATIVE	TRUE NEGATIVE	UNKNOWN

Your task here is to measure and decide how well your BLAST and PSI-BLAST results agree with your gold standard. When comparing predictions with a gold standard "truth", the two most commonly used measures of efficacy are *Coverage* and *Error*. Coverage, in this context, shows what fraction of the known true homologs your method predicted (or how many you missed). It also known as *Sensitivity*, or the *True Positive Rate* (TPR). Error describes how many of your positive predictions are wrong. It is also known as the *False Positive Rate* (FPR). Instead of Error, sometimes the terms *Precision* or *Specificity* are used: $Precision = Specificity = 1 - Error$. Using the table above, you can work out that:

$$Coverage = \frac{TP}{(TP+FN)}$$

$$Error = \frac{FP}{(FP+TN)}$$

Note that it is easy to maximize Coverage, by choosing a method which is more likely to produce false positives than false negatives. Likewise, you can minimize Error with a method more likely to produce false negatives than false positives. Having both a high Coverage and a low Error, however, requires minimizing false positive and false negatives simultaneously, which is substantially more difficult.

Think of how to count the number of TP, FP, FN and TN using the list of BLAST and PSI-BLAST hits you've created in step 2, and the homology calls (similar/different) you've made in step 3.

You will visualize the performance of BLAST and PSI-BLAST in a so-called Receiver Operator Characteristic (ROC) plot. A ROC plot can be made by sorting your (PSI-)BLAST results by their hit score (e-value), and plotting the Coverage vs. the Error for each subsequent entry. Refer to http://wikipedia.org/wiki/Receiver_operating_characteristic for a thorough explanation of what ROC plots are and how they can be used to evaluate, compare, and refine classification methods like ours.

When working with ROC plots, the Area Under the Curve (AUC) is often taken as a measure to evaluate performance. Normally, the area under the curve of a function $f(x)$ can be found by taking the integral:

$$\int_{x_1}^{x_2} f(x)dx = F(x_2) - F(x_1)$$

In our case, we can only approximate $f(x)$, so it is not possible to evaluate the integral algebraically. However, we know $F(x_i)$ for several values of x_i , so we can evaluate the integral numerically. Think of a clever way to do this, and complete the provided skeleton script `create_roc_plot.py`. This script will parse your (PSI-)BLAST and benchmark results, calculate the coordinates for your ROC plot, integrate the AUC, and create the corresponding figure.

[Q4.1] What does the ROC plot of a random method look like? What would the ROC plot of a perfect method look like? What is the AUC for a random and for a perfect method? *[Results & Discussion]*

[Q4.2] Do you see a difference in the ROC plots of your BLAST and PSI-BLAST results? Discuss why (not)! *[Results & Discussion]*

[Q4.3] Which pair of proteins that is classified as different by your benchmarking database has the lowest e-value? Where do you find this pair of proteins in the ROC plot? What is the e-value? Given this e-value, do you think it is likely that these proteins really are not homologous? *[Results & Discussion]*

[Q4.4] Consider your research question. Discuss the advantages and disadvantages of using GO, Pfam or SCOP (All three! Not just your chosen database!) as a gold standard. Do you think that your database of choice is suitable as a gold standard? Include topics like biological relevance, reliability, and the size of the database in your discussion! [Results & Discussion]

[Q4.5] Proteins are grouped on different levels in the three databases. In GO there are terms; in Pfam there are the families and clans; and in SCOP there are families, super-families, folds and classes. What does it mean for your definition of a true homolog, if you start comparing proteins at these different levels? Discuss the differences in how homology is defined in how these three databases are constructed! [Results & Discussion]