<u>CSE 182 Final Project</u>

Mustafa Guler and Pranav Sujitkumar

Introduction and Motivation

Although advancements in medicine help immensely in the fight versus disease and bacteria, many bacteria have been able to gain resistance to previously successful medication. One bacterium that has developed resistance is an opportunistic bacterium found in hospitals called Acinetobacter baumannii. An opportunistic bacterium takes advantage of situations that are not normally present, such as weakened immune systems of patients within hospitals. When these patients are treated with drugs, some bacteria manage to survive because of newly developed drug resistance, causing the next generation to become more resistant as a whole. Today, Acinetobacter baumannii has many drug resistant strains due to heavy drug resistance gained within hospitals. Because the annotation of Acinetobacter baumannii is very poor, it is tough to identify what proteins may cause this increased drug resistance. Thus, the motivation behind our project was to query different protein databases with 100 unknown sequences (151-250 in UP000006737.fasta), and assign function to see what proteins may take part in drug resistance or could be novel targets for new antibiotics.

Methods

General Approach

The four protein databases we wanted data from were Prosite, Pfam, BLAST, and PRINTS. The original approach was to use the different databases' local versions of their tools wrapped in python scripts in order to execute the programs and parse the data. Later, the modules written for each database were combined into a controller which ran all of them in parallel and outputted data in a TSV format. Auxiliary scripts were written to convert this TSV format to an indexed JSON format and to query our hand assigned function against the GO-Slim database. The data tables can be found in the root/Results/ directory.

Prosite

root/Prosite/prosite.py

Prosite is a regular expression based database that uses predefined regular expressions to find patterns within query sequences. Each of these patterns is assigned a general function and can be used to find the function of a portion of a protein.

Prosite's local tool (ps_scan), is a Perl script that searches against the local Prosite database (Prosite.dat). This database was stored in the databases directory in our project root folder. Both the tool and the database were found on the Prosite ftp servers. The ps_scan.pl perl script made use of two flags we used to generate a three tiered calling system. The first flag "-s", excluded all high probability matches from the results. The second flag "-l", could be modified to allow for a less confident result by redefining Prosite's "cutoff parameter". For each sequence, Prosite was called with high probability matches excluded, and the original Prosite cutoff parameter. If this resulted in no matches, the same sequence was queried again with a lower cutoff parameter. If this second query also got no matches, the sequence was queried for a third and final time with high probability matches included but the original Prosite cutoff parameters. Based on which one of these calls were successful, we assigned our own confidence values to the results of the query (1-3, with 3 being the strictest parameters in the call to Prosite).

The data reported from Prosite was included in the Prosite hits, Prosite confidence, and Prosite codes columns in the data tables. The Prosite hits column contained the assigned function from Prosite. The Prosite confidence contained our assigned confidence level based on the parameters when the sequence was queried. The Prosite codes column contained the Prosite accession numbers for each of the Prosite results in the Prosite hits column. The Prosite hits and the Prosite codes were extracted from the local tool's results using regular expressions. This module, despite being a three-tiered approach was quite fast even in the worst case.

Pfam

root/Pfam/pfam.py

The Pfam database is a large collection of protein families, each represented by multiple sequence alignments and hidden Markov models (HMMs).

The PfamScan tool, which can be found on the Pfam ftp servers, is based on the Hidden Markov Model search of HMMER. This local tool is also built around a Perl script that was edited slightly to allow for higher probability. The Pfam scan tool uses Perl modules that were included with the download but would only be found if the tool was run within the folder containing it. The pf_scan.pl was modified to specify the actual path to the Perl modules that it relied upon. All the databases that were required for the search were packaged with the tool download, but it required binaries from HMMER to run correctly. Additionally, Pfam scan was reliant on a CPAN module called Moose which was installed using the command line interface for CPAN. The Pfam scan module was designed in a similar fashion to the Prosite module except there were no tiered callings.

The data reported from Pfam were Pfam Hits, Pfam Codes, Pfam Clan Names, Pfam Clan Codes, and Pfam E-vals columns. The Pfam hits column contained information about the protein domain families, while the Pfam codes column contained the Pfam accession numbers for each of the Pfam hits. The Pfam Clan Names column contained the name of the clan the Pfam hits were associated with. Pfam clans are a collection of Pfam entries which are related by similarity of sequence, structure or profile-HMM. Just because a Pfam hit was obtained, there is no guarantee it will belong to a clan. The Pfam E-vals column was the Pfam e-value associated with each Pfam hit. The relevant data from the Pfam results were extracted using regular expressions. The Pfam module was noticeably faster than the Prosite module even if the Prosite module succeeded of the first call.

PRINTS

root/Prints/prints.py

PRINTS is a database that uses a group of conserved motifs used to characterize a protein family. What sets it apart from other databases of a similar nature is that it can encode protein folds and functionalities using information from motif neighbors as well.

Instead of using a local tool, we used the PRINTS fastBLAST online tool built into the PRINTs website to search by sequence. fastBLAST as the name implies, is a faster version of BLAST that searches for near exact matches against a smaller database. This tool required a different approach than we had been using previously. Instead of downloading a command line interface pipe tool, we used the selenium library for python using a headless browser called phantomJS to submit queries online. Selenium is a prototyping tool for building test scripts. This is primarily used to test website functionality, but can just as easily be used to parse websites by submitting queries and scraping data. The PRINTS module we wrote while also written in python didn’t explicitly separate the running of the tool and extraction of data.

The data returned from PRINTs were Prints hits, and Prints codes. The Prints hits column contained the resulting fingerprints from the PRINTS query. The Prints codes column contained codes for each of the hits in the Prints hits column. The Prints codes were extracted directly from the resulting webpage after the query, but the Prints names required following a link on the results page and parsing the HTML code with a regular expression. PRINTS was surprisingly found to be the fastest module, despite having to query online instead of locally like the other tools.

Blast

root/Blast/blast.py

BLAST is an alignment tool that locally aligns query sequences against its database. The database we used was NR, which contained non-redundant sequences from GenBank translations and a few other databanks.

Even while restricting ourselves to using the local BLAST tool available on the BLAST ftp server under the ncbi-blast+ package, there were many options for how to structure the module. The first of which was deciding what BLAST call to use. We decided to use the blastp tool instead of psiblast or deltablast which also work against proteins. This is because PSI-BLAST and DELTA-BLAST were slower even though they were more sensitive. However, sensitivity difference between these different BLAST calls proved to be less significant factor for module construction that the running time. After we chose blastp as our tool, we had the option of running it locally against a downloaded version of the nr database or submitting to the BLAST server using the –remote option. Neither of these options proved to be particularly fast compared to the other modules. When the local database version was running using eight cores, it did not compare favorably to submitting queries to the BLAST servers so the –remote flag was used. However, this proved to be a problem when parallelizing the calls to BLAST as the BLAST servers failed to respond when multiple queries were submitted at once from different threads. This forced us to run each BLAST call in series which significantly slowed down our running time and was easily the rate limiting factor. Our first solution to this problem was to run all of our BLAST calls overnight and save the results pages, parsing them at a later time to avoid having to generate the results file each time. While this proved useful in developing the tool, it did not seem viable as a long term solution. A solution we came up with was to use selenium to query the BLAST database and scrape the search ID number, periodically checking for the job's completion using the BLAST search ID capability. Additionally, instead of using blastp when doing the web version of our module, quickBLASTP was used. quickBLASTP adds a preprocessing to the nr database so that it can run faster but is limited to the nr database as of right now. Although this faster web scraping version of BLAST was implemented and observed to be much faster (three minutes to less than 30 seconds), it was not included in our final data generation pipeline due to time constraints in passing data along to the other groups.

The data reported from BLAST was included in the Blast hits and Blast E-vals columns. The BLAST hits column contained the names from the top ten (if there were 10 hits) BLAST alignment results. The Blast E-vals column contained the e-values corresponding to each hit from the Blast Hits column. The raw results from the initial approach were saved under the format accession#_raw.txt under the blast_hits directory.

Function and GO:slim Assignments

root/ExtraFiles/extract_go.py

Functions for each protein were hand assigned based on the results from the previously listed databases and contained in the Functions column. The results were searched online to get more info and in the case of low confidence or no resulting hits, more research was done into previous studies of similar proteins. Sometimes, the BLAST results were all “hypothetical protein”. In this case, we did a tblastn search, which used the nt database translated into amino acids and searched the query against this translated database. This is helpful as it can get hits specifically on the original nucleotide genome database rather than trying to align with sequences of biological protein function. We tried to keep the function assignment as general as possible without losing important specifics in order to accommodate for easier GO:slim assignment. Many times, the hits from the different databases complemented each other, creating a more holistic view of the protein and its functions. The specifics from the research done before assigning each protein’s function was put in another column called Comments.

The GO:slim script is located under root/ExtraFiles/goslim.py. The script prints out exact matches to the function we assigned or synonyms to those functions such as ligase and recombinase (in some sense at least). Looking at the printed out matches, we assigned GO:slim codes to each function. However, there were cases of no GO:slim matches. In this case, we manually looked through the GO:slim databases after thinking of even more general names until we found a result.

Proteins of Interest

There are a variety of interesting proteins that we found when annotating unknown proteins 151-250 that may show why Acinetobacter baumannii has such good antibiotic resistance and where it may be susceptible to drug treatments. Since the proteins are unknown, they will be referred to by their accession number.

The first interesting protein is ABO11958. The hits from the protein databases were examined first. Prosite returned a hit with the highest level of confidence that the protein contained a sirtuin catalytic domain. This domain is involved in NAD-dependent protein deacetylation. Since DNA is tightly wound around groups of histones, transcription of DNA occurs once the histones are acetylated causing a transformation to make the accessible to proteins involved in transcription. Therefore, these deacetylases are targeted against histones and reduce transcription. Pfam returned Sir2 family (a family of deacetylases known in full as silent mating type information regulation) with an e-value of $1.4e^{-45}$. This family contain a sirtuin catalytic domain and is in direct agreement with the Prosite results. In addition, BLAST returned NAD-dependent deacetylase with an e-value of $3e^{-174}$, supporting the Pfam results as sir2 family proteins use nucleotide cofactors. Finally, the last database, PRINTS, returned FAD-dependent pyridine nucleotide reductase, which points again to the sir2 family which use these nucleotide cofactors to function as deacetylases. When thinking about the function of a deacetylase in the context of antibiotic resistance, it seems odd that a deacetylase could help an opportunistic bacterium as it is rendered unable to replicate its DNA to the fullest capacity. However, sirtuins also play key roles in other processes as well, notably stress resistance and preventing apoptosis. Though they have some function as tumor suppressors, they are bifunctional and have been implicated in the initial progression of tumors and helping tumors gain resistance against treatment. This constant resistance can be gained if the deacetylases mutate and have high amounts of activity within the cell. According to Alexandre Vendrell and team in their study about histone deacetylases, Sir2 deacetylases prevent programmed cell death by sustained activation of the Hog1 stress –activated protein kinase in yeast. Since the Sir2 family is apparent in many organisms, it is reasonable to think it may function similarly in Acinetobacter baumannii. Overall, sirtuins act differently given the context of their environment, but their ability to confer constant resistance to cell stresses can play a large part in Acinetobacter baumannii gaining resistance to constant treatments faced in hospitals.

The next protein to observe is ABO13685. Prosite hit was of the highest confidence level and returned that the protein contained a GGDEF domain, an almost identical hit to BLAST, which returned it with an e-value of 0. This domain is named after the conserved central sequence pattern GG[DE][DE]F and is widespread in prokaryotes. However, it is most often found in regulatory domains and is involved in the catalysis of cyclic diguanylate. Diguanylate cyclase happened to be the top hit from Pfam with an e-value of $1.6e^{-39}$, while the Pfam clan name was nucleotide cyclase superfamily under which the GGDEF domain exists. Lastly, PRINTS returned a cell division protein FtsZ signature. Diguanylate cyclase is involved in cell division as an inhibitor during reductive and environmental stresses. There are multiple ways that cyclic diguanylate can help give drug resistance. The first is from the hits given from the protein databases. Although diguanylate cyclase inhibits cell division and proliferation of the bacterium, it still confers resistance during environmental stresses. The second way is related to the pathogenicity of the bacteria as well. Cyclic di-GMP is involved as a secondary messenger essential in biofilm formation. Biofilm are group of microorganisms in which cells stick to each other and adhere to a surface and are often implicated in chronic diseases, similar to the type of diseases Acinetobacter baumannii transmits. If an infection develops biofilm, it becomes much harder to treat and since biofilm cause differential regulation of genes to become more resistant to antibiotics and the body's own host defenses. Research has shown that antibiotics against biofilm sometimes induce extracellular DNA release, allowing the bacteria to live on elsewhere if its DNA is taken up. However, this provides a good chance to try to treat this bacterium since cyclic di-GMP is needed as a messenger. Therefore, by using enzymes that inhibit diguanylate cyclase which synthesizes the secondary messenger c-di-GMP, we can hope to reduce antibiotic resistance in the cell by preventing biofilm formation.

A protein heavily involved in this biofilm formation is from the iron uptake mechanism AB012980. According to Pfam this protein is siderophore-interacting protein under the Pfam clan ferredoxin with an e-value of $2.3e^{-17}$. BLAST returned the exact same hit with an e-value of $1e^{-148}$, however, it noted the hit was from the Acinetobacter baumannii strain. PRINTS showed a disease resistance protein signature (however in the context of bacteria it seems to mean antibiotic resistance). In the paper "Growth of *Acinetobacter baumannii* in Pellicle Enhanced the Expression of Potential Virulence Factors" by Sara Marti and team, the upregulation of these siderophore proteins promotes the growth of biofilm due to the ferric ions that were just taken up. The iron ions seem to provide antibiotic resistance and help in biofilm production. The paper also states that the overexpression of lipase and a long chain fatty acid help recycle lipids inside the biofilm. One of the unknown proteins, ABO11565, is a prokaryotic membrane lipid attachment site according to Prosite. There were a few other proteins with similar hits from Pfam, but this one has relation with the recycling of lipid function discussed in the paper. This protein is part of the permease family according to Pfam (e-value $5.4e^{-10}$), and has a permease domain according to BLAST (e-value $1e^{-53}$). PRINTS returned a GPCR superfamily signature which has another interesting connection to Acinetobacter baumannii. According to Julia Stahl and group in their paper "*Acinetobacter baumannii* Virulence Is Mediated by the Concerted Action of Three Phospholipases D", the PLD mediated cleavage of lysophosphatidylcholine (LPC) triggers signaling cascades via G-protein-coupled receptors in host cells. They argue each PLD (phospholipase D) molecule has a factor in contributing to virulence. Therefore, the lipid attachment site should be of importance to Acinetobacter baumannii. Overall the lipid receptor and iron uptake proteins functions found among the unknown protein seem to also have a key role in biofilm production, and therefore antibiotic resistance. Thus, targeting the receptor may have favorable results in being able to target the bacterium with antibiotics.

Another protein to consider, ABO11928, has involvement with how the phage integrates its DNA with its host. The Prosite hit returned with the lowest confidence, thus its matches included very high probability hits and general functions such as N-glycosylation site, protein kinase phosphorylation site, etc. However, the rest of the databases gave confident and related results. Pfam returned phage integrase family (e-value $1.1e^{-6}$), a subset of the Pfam clan DNA breaking-rejoining enzyme superfamily. BLAST returned phage integrase (specifically from Acinetobacter baumannii) with an e-value of $2e^{-138}$, very similarly to the Pfam hits. The PRINTS hits returned the DNAj domain signature. DNAj is a member of the hsp40 family of molecular chaperones, which is also called the J-protein family, the members of which regulate the activity of hsp70s. Despite the PRINTS hit being more involved in protein folding, we still believe the protein is an integrase due to the confidence in blast and Pfam hits. Regardless, the integrin (bacterial version of integrase) function of the protein is essentially for the survival of the bacteria. It is able to integrate its DNA into the host genome and have the host cell replicate and express the proteins from the bacterium. In addition, the integrated DNA can remain dormant in the genome as well, avoiding detection from antibiotics. Previous research has showed that bacterial strains carrying these integrons were significantly more resistant to antibiotics. Integrons can do this by specifically integrating site specific antibiotic-resistant gene cassettes. Though there are many integrons present in pathogenic bacteria, the ones that confer antibiotic resistance seem to be viable targets to inhibit as they can make previously existing antibiotics become more useful as well.

Conclusion

Overall, we believed that our pipeline was fairly successful outside of the speed of the blast module. In future iterations or later improvements on this project we would like to automate the assignment of functions and comments and focus on using web-scraping as our primary design. Also, GO:slim assignment was done in a very naive way and could be improved greatly. A suffix trie to find partial word matches in either the GO name, synonym, or description would have been far more effective and sensitive than the exact search that was implemented. This greater sensitivity could also help with potential automation of function assignment by creating a well-defined set of terms that can be used for function. However, doing such an automation would likely require the parsing of PubMed papers which could end up being very computationally expensive. The speeding up of the blast module was already done in java, along with a GUI for search and results, but has not been included due to its lack of asynchronous programming at this time. If we did have to hand assign proteins, it may have been easier to think about how specific we could be in assigning function given the parameters of the small GO:slim database. In regards to the protein analysis, it is clear that Acinetobacter baumannii have many resistances against bacteria that have made them quite powerful. However, looking at the resulting proteins, it is possible to create antibiotics for novel targets in hope of weakening the bacterium across all strains and progress in the fight versus Acinetobacter baumannii.

Works Cited

Marti, Sara, Yassine Nait Chabane, Stéphane Alexandre, Laurent Coquet, Jordi Vila, Thierry Jouenne, and Emmanuelle Dé. "Growth of Acinetobacter baumannii in Pellicle Enhanced the Expression of Potential Virulence Factors." *PLOS ONE*. Public Library of Science, n.d. Web. 11 June 2017.

Stahl, Julia, Holger Bergmann, Stephan Göttig, Ingo Ebersberger, and Beate Averhoff. "*Acinetobacter baumannii* Virulence Is Mediated by the Concerted Action of Three Phospholipases D." *PLoS ONE*. Public Library of Science, 2015. Web. 11 June 2017.

Vendrell, Alexandre, Mar Martínez-Pastor, Alberto González-Novo, Amparo Pascual-Ahuir, David A. Sinclair, Markus Proft, and Francesc Posas. "Sir2 histone deacetylase prevents programmed cell death caused by sustained activation of the Hog1 stress-activated protein kinase." *EMBO Reports*. Nature Publishing Group, Oct. 2011. Web. 11 June 2017.