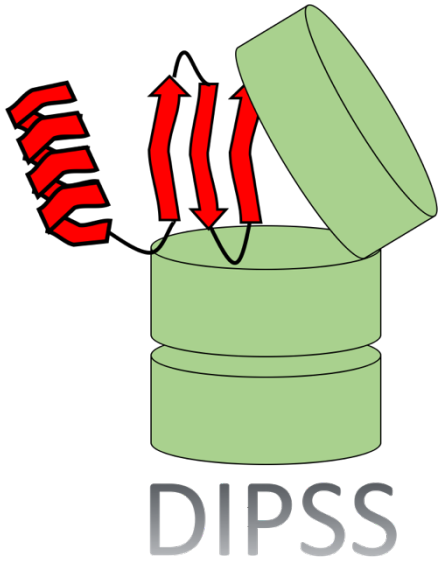
DIPSS: Database for Inquiry into Protein Secondary Structure

Dreycey Albin
COMP 533
Introduction to Database Systems

Abstract

While there are multiple database systems for proteins, many are difficult to locate, and most do not integrate multiple databases. Furthermore, the available databases online restrict user access, disallowing end users from running queries. To circumvent the need of a more flexible protein-based database, here the Database for Inquiry into Protein Secondary Structure (DIPSS) is provided as a user-friendly solution. The DIPSS database integrates information from UniProt, UniRef, Protein Data Bank (PDB), the Gene Ontology (GO), and also includes two curated datasets from Kaggle. In addition, DIPPS also has SQL based functions giving users the ability to predict the molecular and cellular functions of input proteins, as well predict the potential secondary structure for an input protein.



Problem Description

Proteins can be described as macromolecular units that carry out majority of the functions needed to sustain life. These polymeric units are made up of smaller monomeric units called amino acids, each of which has a unique chemical moiety which defines each particular amino acid. There are 20 common naturally occurring amino acids that are used in a multitude of combinations to construct larger polypeptides (or proteins). This results in the following number of combinations for a peptide sequence:

$$Peptide\ Sequence\ Combinations = 20^N,$$
$$Where\ N\ is\ the\ number\ of\ amino\ acids$$

Therefore, the number of potential combinations increases greatly as the length of the underlying protein is increased. For example, for a typical protein of 70 amino acids, this amounts to more combinations than there are stars in the milky way. However, nature has only used a small subset of these possible combinations to define the proteins found in the living organisms that have been studied.

Adding to the astonishing number of potential sequences is the number of structures resulting from the underlying sequences. The structure of a protein is what determines the function of the protein, thereby following the traditional idea in biology that the structure and shape dictates the function. These are crucial concepts in understanding how to predict the both the structure and function of proteins, and how to reverse the protein structure prediction problem in effort to systematically design proteins.

Protein design is an important goal, as there are many distinct industries that could benefit from the ability to accurately designing proteins. Proteins have the ability to turn chemical energy into physical energy, and vice versa, giving them the winning spot for efficient molecular machines. Examples of this include turning chemical energy into work, as seen in the contraction of muscles, or using pH gradients in order to create chemical bonds within the mitochondria (Huang, Boyken, Baker, 2016). It is for these reasons that the community would benefit from having a database that integrates the wide plethora of information regarding proteins, in effort to more effectively query for consistent and usable information on proteins. To solve this unmet need, this research outlines a database for protein-based information. **This database, referred to as *Database for Inquiry into Protein Secondary Structure* (DIPSS), aims to integrate structural information from the Protein Data Bank (PDB) with other widely used database systems, such as UniProt and the Gene Ontology (GO)**.

Data

To find datasets relating to bioinformatics and proteins, Kaggle was searched for datasets relating to biological information. This search resulted in the finding of two separate datasets containing pertinent information on protein structure: (1) “Protein Secondary Structure”: “*Curated dataset for protein secondary structure prediction*”; and (2) “Structural Protein Sequences”: “*Sequence and metadata for various protein structures*”.

(*www.kaggle.com/alfrandom/protein-secondary-structure*; *www.kaggle.com/shahir/protein-data-set*, respectively).

The former dataset allows for the relationship between secondary structure and primary protein sequence to be related. Within this dataset, the protein sequences are derived from the 3-dimensional crystal structures residing in the PDB, and are thereafter translated into secondary structure categories. There are two separate secondary structure naming categories used in this dataset: (1) sst3 (a 3-state description of the protein secondary structure); and (2) sst8 (an 8-state descriptor of the secondary structure for the protein). The eight-state descriptor contains, theoretically, the most information on the secondary structure. These consist of the following states: C = irregular loops/random coil, E = β-strand, H=α-helix, B= β-bridge, G=3-helix, I: π-helix, T= Turn, and lastly, S= Bend. These abbreviations are condensed for the sst3 descriptors in the following way: E (containing E, B), H (containing H, G, I), and C (containing C, S, T). These secondary structures are calculated from the DSSP program (Kabsch, Sander, 1983).

The latter dataset contains information about different structures deposited into the PDB. The original data for this dataset was retrieved from the Research Collaboratory for Structural Bioinformatics (RCSB). While this dataset could be reconstructed using the available information from the RCSB website, having the curated .csv file from kaggle helps streamline the process, since the downloaded files from the RCSB would have to be denormalized in effort to get the same dataset all in one table (Burley et al., 2019).

Database Systems

In effort to gather more pertinent information on the proteins within the Kaggle datasets, UniProt was used to download all available information on *human* proteins. The original download included a combination of attributes for all of the proteins, which UniProtKB allows the user to choose (The UniProt Consortium, 2019). In this regard, UniProtKB allows the user to manually construct a dataset of choice and subsequently download it as a .csv (***Figure 1***).

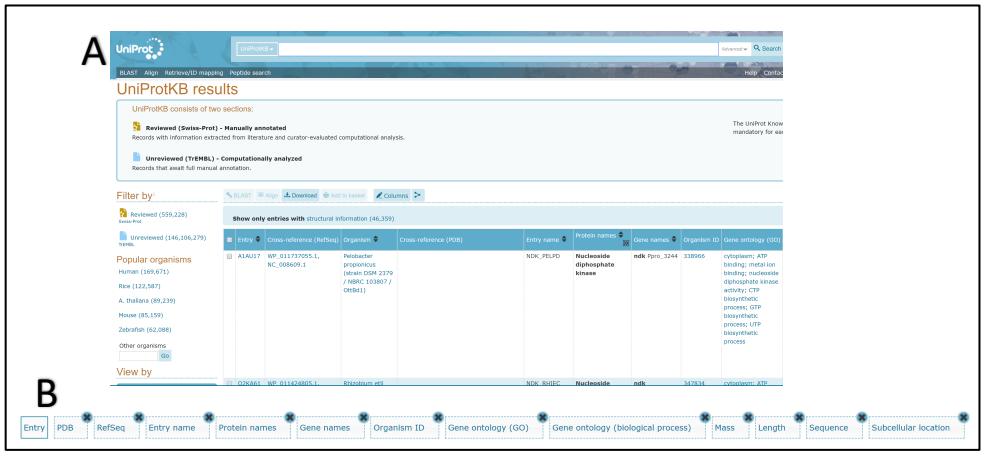


Figure 1- Screenshot of the UniProtKB search screen, where users are able to retrieve desired information about proteins. (A) a screen shot of the results from a specified query; (B) screenshot of the manually selected columns accompanying a UniProtKB query.

Team

The team for the database implementation, declarative SQL, imperative SQL, and the making of this manuscript consisted solely of the primary author, Dreycey Albin.

Design

Entity Relationship Diagram

The Entity Relationship Diagram (ERD) for the DIPSS database was designed using Chen notation (***Figure 2***). After revisiting the original design implementation, the ERD seen in Figure 2 was used to help guide the normalization of the original implementation. This led to the successful data renovation used to implement the database in First Normal Form (1NF).

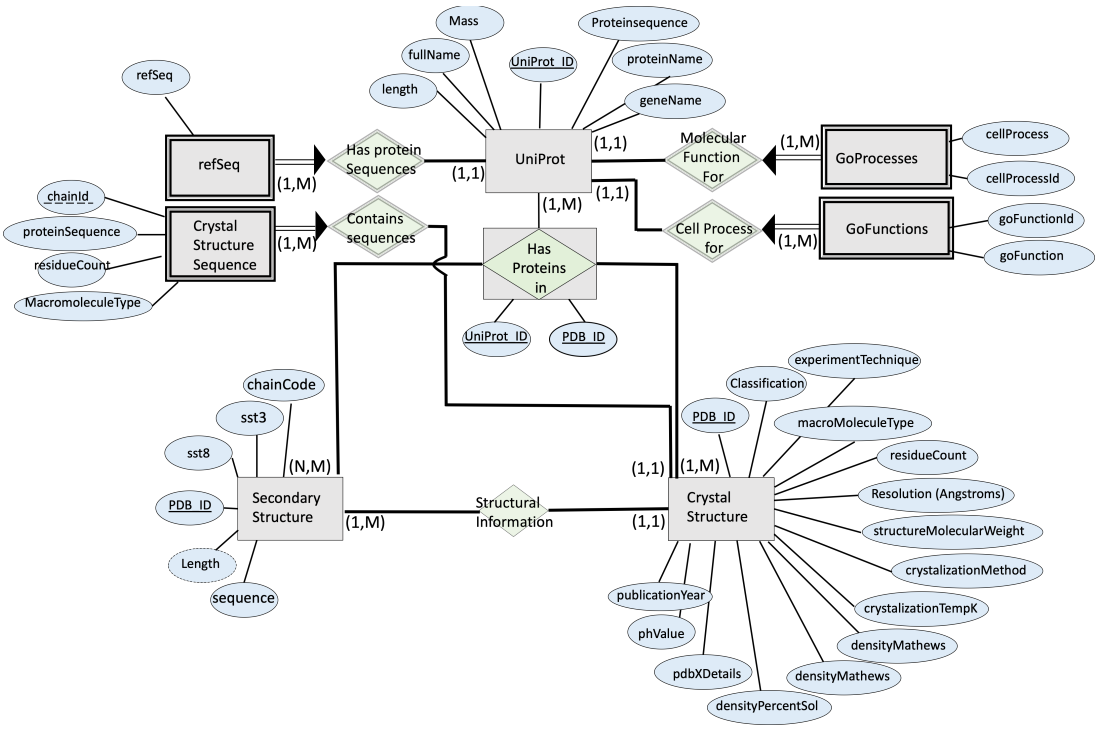


Figure 2- Entity Relationship Diagram for the DIPSS database featured using Chen notation. The tables constructed in the database are seen in grey, consisting of both entities and associative entities. Relationships are featured in transparent green, and the attributes for the corresponding entities are colored light blue. The illustration was made using Microsoft PowerPoint.

Entity Descriptions

The refSeq entity contains the information regarding gene names in the RefSeq database. This was obtained using the UniProtKB search engine as explained in the "Database Systems" section. This table has three attributes: (1) Foreign Key relating back to the UniProt table; (2) The corresponding RefSeq ID; and (3) the Chain ID for that particular protein-gene combination. Having the RefSeq information is extremely valuable for future advancements on the database, as it allows for the integration of DNA sequence information from the RefSeq database.

The UniProt entity has information regarding the human protein sequences. This entity is the primary entity that contains information on the protein sequences being queried in the DIPSS database. In many respects, it can be thought of the entity relating the Kaggle datasets to other largely used protein databases (namely Gene Ontology (GO) and RefSeq). In addition, the UniProt entity has general information about the proteins, such as the sequence length, gene name, and mass.

Both weak entities with GO process information relate the PDB references back to known functions for the proteins. These are designed to be weak entities for the UniProt entity, since protein may have many functions, a term called "*pleiotropy*". By design, this indicates that the PDB ID is used as a foreign key to relate back to the original UniProt table. Each PDB ID is then matched to either known cellular process, in the GoProcesses weak entity, or to known molecular functions, in the GoFunctions weak entity. Besides relating back to the corresponding PDB ID, these weak entities also contain GO IDs. The GO IDs allow for more information to be integrated into the database during future additions, and if output to a query is saved, allows users to find more information regarding the GO ID. Also, there is an attribute for English description for each GO ID, which in turn describes either the associated cellular process or molecular function. An example of a description for the GoFunction table would be "execution phase of apoptosis" corresponding to the GO ID "GO:009719".

The Secondary Structure entity corresponds to the table containing information on the secondary structures for the proteins. This table was made by a direct upload of the secondary structure dataset retrieved from Kaggle. It contains information on the secondary structure of the proteins, for both sst3 and sst8, calculated from the DSSP program. In addition, this entity gathers information about the chain code, length of the sequence, and the chain sequence.

As for the Crystal Structure entity, this hold information regarding to crystal structure information about the proteins in the PDB. This entity has information from one of the datasets retrieved from the Kaggle crystal structure dataset repository. The attributes in this dataset are consistent, and give information about the structures residing in the PDB. This information ranges from data about the method used for obtaining the structure(s) to metadata retrieved on the structures, such as the publication year and resolution.

In conjunction with the Crystal Structures entity is the Crystal Structure Sequence weak entity. This entity has data from a second dataset provided by the crystal structure Kaggle repository. The major difference here is that this dataset splits the structures into the underlying chains, whereas the Crystal Structure entity is per PDB ID (where each PDB ID may contain many chains). This information is unique in the database, as some of the proteins in the PDB also contain DNA/RNA, and some of structures are solely of DNA or RNA.

Example Use Cases

There multiple potential use cases for the DIPSS database. It was designed with multi functionality in mind, and it serves to meet this criterion. An example would be if a protein sequence was obtained and one wanted to characterize its potential function. With a heuristic function for figuring this out already implemented in the current DIPSS database implementation, this is possible. A different use case could be for dataset construction: machine learning algorithms are continually being used to decipher secondary structure in proteins, and this database could be used to curate training datasets for this task. These datasets could be filtered based on molecular or cellular function, or better yet, these datasets could be filtered based on the resolution of the structure imported into the PDB. Using these filtered datasets for training the machine learning algorithm could help an investigator obtain a more accurate model. One final interesting use case could be for the molecular characterization of protein k-mers (substrings of the larger protein sequence). It may be that certain k-mers are commonly associated with certain molecular functions, and the DIPSS databases is uniquely posed to elucidate which k-mers correspond most frequently to what functions and secondary structures (***Figure 3***).

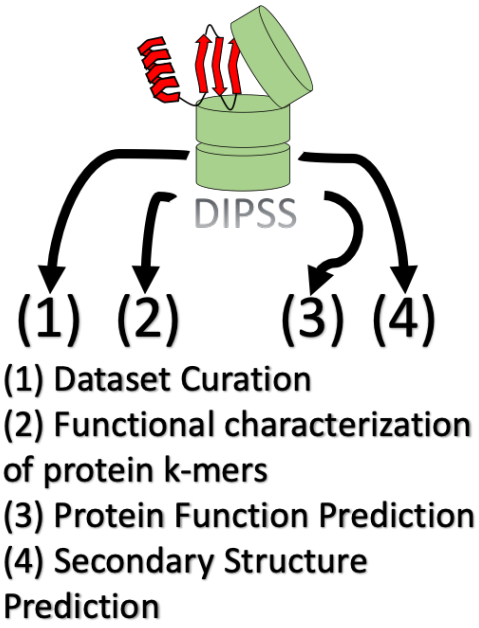


Figure 3- Outline of the different use cases for the DIPSS database.

Methods

There were several steps followed in order to both prepare the data and to design functions for the analysis of the data in the DIPSS database. Each step wrapped into the methods was essential for building the DIPSS database, ensuring that the database implementation had all needed tables and functions necessary for revealing the full potential of the database.

Uploading The Datasets into Postgres

Uploading the datasets into Postgres was relatively straight forward, however, the there was optimization involved in determining the data types for each of the attributes. Each of the datasets consisted of a file in .csv file format, which allowed for an easy upload into Postgres using the following command for each csv once a corresponding table was made:

$$COPY < TableName > FROM\ 'PathToFile.csv'\ WITH\ (FORMAT\ csv);$$

An example of this process can be seen in ***Figure 4***, where a table is first made before the data is imported into the table.

```
CREATE TABLE GOPROCESSES (
    UNIPROT_ID VARCHAR(3000),
    CELLPROCESSID VARCHAR(3000),
    CELLPROCESS VARCHAR(3000),
    FOREIGN KEY (UNIPROT_ID) REFERENCES UNIPROT(UNIPROT_ID),
    PRIMARY KEY (UNIPROT_ID, CELLPROCESSID));

COPY gofunctions FROM
'/Users/da39/Desktop/CLASS_WORK/building_a_database/curated_csvs/functionTable.csv'
WITH (FORMAT csv);
```

Figure 4- Screenshot of the commands used to create a table titled "GOPROCESSES" and subsequently load the data in the corresponding csv file.

Curating the Database For 1st Normal Form

The original database schema lacked a robust form, and several attributes in the schema were multivalued, causing the database to be denormalized (***Figure 5***). In addition, the original database configuration lacked the ability to map back to UniRef sequences. While the UniRef entity is a seemingly small addition to the overall schema, going from Figure 5 to Figure 2, adding this table allows for the integration of the UniRef database in future editions.

The UniRef database was added by using the UniProtKB resource, selecting matching PDB IDs to the corresponding UniRef IDs, and only selecting the sequences for humans. Since some PDB IDs have multiple UniRef IDs in this dataset, this therefore represents a weak entity

of the larger UniProt entity. Uploading this as a weak entity worked as expected, where no problems occurred when using the PDB IDs in this table as a foreign key to the UniProt entity.

To get the database in first normal form (1NF), one major objective was to split the GO attributes into weak entities. The first schema for the database treated these attributes as multi valued attributes, however, this is not sufficient for having the database in 1NF. The original plan was to modify the attribute in the SQL database, but this proved to be extremely difficult as certain attributes contained inconstancies for teasing apart the descriptions from the GO IDs. Because of these difficulties using SQL to parse these attributes, a python script was made to parse the original UniProt GO attributes. This python script splits the description and GO ID, relating each back to the corresponding PDB ID. The output for this returns a csv formatted file that can directly be imported into the Postgres DIPSS database as shown in Figure 4.

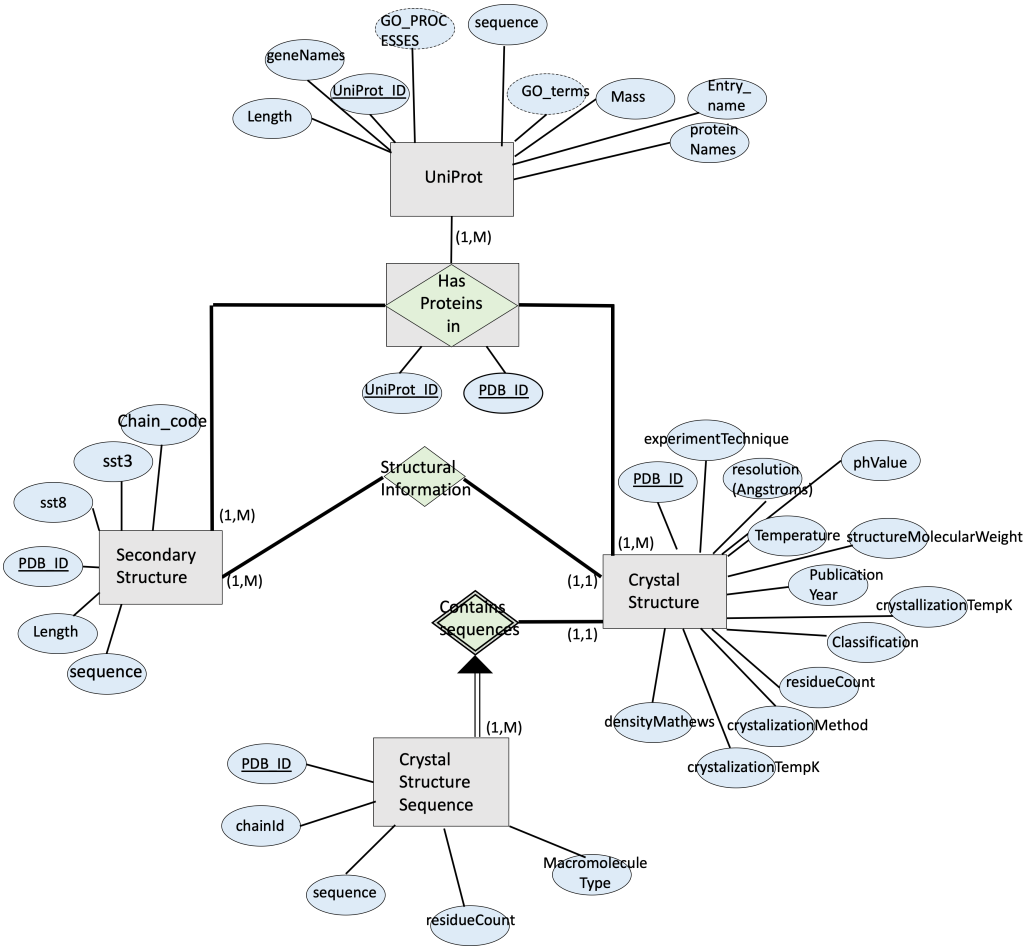


Figure 5- The ERD corresponding to the original database schema before normalization. Tables/entities are shown in grey, attributes in blue, and relationships in green. The illustration was made using Microsoft PowerPoint.

Checking for Consistency

Before uploading all of the CSV files into the database, consistency was ensured by manually evaluating all of the attributes for each entity. Consistency was ensured by evaluating the format used for shared attributes between different entities, such as the format used for the PDB IDs. Fortunately, this manual evaluation showed that the attributes used a similar format for all of the shared attributes between the separate datasets. Likewise, the validated consistency for the GO IDs and UniRef IDs was manually confirmed, making the process of incorporating other databases into DIPSS trivial.

Declarative SQL

A series of declarative SQL queries were used to test the ability of the DIPSS database to deliver answers to protein structure related questions. The major focus for the declarative SQL queries was to make sure that the queries were meaningful to end users; something that a user of the database would be interested in asking. These queries were thereafter evaluated on the ability to produce valid answers (or what would be expected, assuming the database contains enough information). This consisted of making sure that the queries gave expected results, such as the number years of documented structures ranging back to the 1960s.

Imperative SQL

As in the case for the declarative SQL queries, the primary goal for the imperative SQL commands was focused on producing functions that an end user may actually find valuable for the DIPSS database. After the normalization of the DIPSS database, there were many functions that could be implemented, however the goal for the first version of DIPSS was to implement algorithmically challenging functions. This is because challenging functions were thought to test the ability for the database to produce novel results, as in comparison to basic queries that may be used on the entities separately. In addition, building the challenging functions first works well as a proof-of-concept for the ability for DIPSS to produce novel results. After implementing the algorithms in imperative SQL, test cases were used to evaluate the ability for the implemented functions to produce reliable information.

Deliverables

Tables

The tables for DIPSS were designed with the database schema in mind. Each of the tables were named in accordance with their counterpart entity name in the ERD (Table 1). The foreign and primary keys for the database were also chosen with regard to the schema, making sure that the database accurately reflected the outlined ERD in Figure 2.

Table 1- Summary of the tables within the DIPSS database.

Table Outline in DIPSS				
Table Name:	**Number of Rows:**	**Number of Attributes:**	**Corresponding Name in the ERD:**	**ERD Type:**
HASPROTEINS	52246	2	has Proteins in	Associative Entity
UNIPROT	169675	7	UniProt	Entity
SECONDARYSTRUCTURE	393732	6	Secondary Structure	Entity
CRYSTALSTRUCTURESEQUENCE	467304	5	Crystal Structure Seqeunce	Weak Entity
CRYSTALSTRUCTURE	140911	14	Crystal Structure	Entity
GOPROCESSES	209707	3	GoProcesses	Weak Entity
GOFUNCTIONS	523817	3	GoFunctions	Weak Entity
REFSEQ	80581	2	refSeq	Entity

Key Queries

As described in the methods section, the major declarative SQL queries were decided upon based on what a DIPSS user would find useful. A clear example of this is how many PDB structures were uploaded after or before a specified date. Because techniques to elucidate macromolecular structures are continually improving, it may be assumed that the number of crystal structures increases each decade. Using DIPSS, this stated inquiry can indeed be answered, showing that the number of structures elucidated does increase each decade (***Figure 6***). Furthermore, it can be seen that the current decade has already out produced the previous decade, even though the database only has the years up to 2018. Furthermore, the database can also be used to make mathematical models. To illustrate this point, ***equation 1*** represents a

fourth degree series expansion used as a rough predictor for the number of structures per decade. This give a rough model for the number of structures expected each decade, but serves well to illustrate the fact that simple DIPSS database queries may be used to build models. Overall, there are a number of queries that were run on the DIPSS database. These declarative SQL queries serve to show the schema of the DIPSS database supports user interests, and allows for complex questions in regard to protein structures to be answered with relative ease.

A

```
SELECT COUNT(*)
FROM CrystalStructure CS
WHERE publicationyear < 1970
AND  publicationyear > 1960
AND NOT publicationyear=-1
```

B

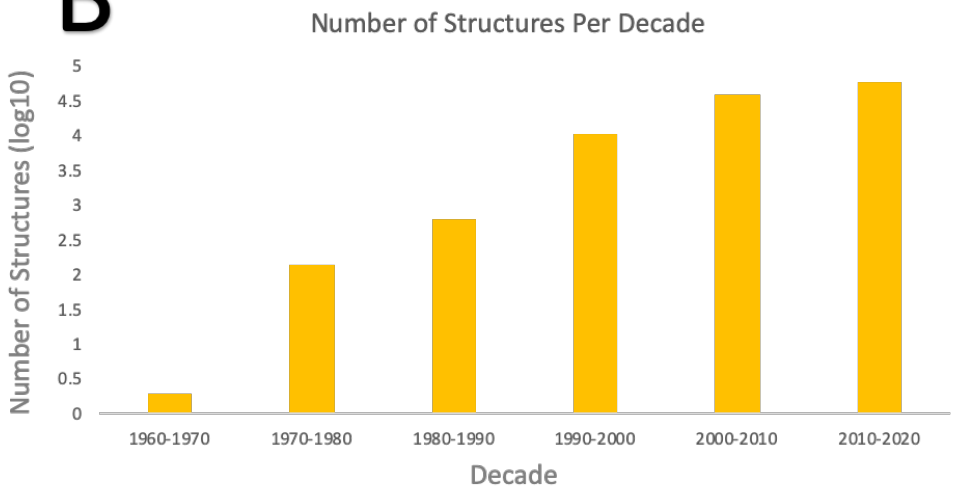


Figure 6- Declarative SQL used to evaluate the number of structures over time. (A) The query used for the inquiry in the DIPSS database; (B) Log scaled bar plot showing the number of structures per decade in the DIPSS database.

$$F(x=\text{decade}) = -767.06\left(\frac{(x-1960)}{10}\right)^4 + 10521\left(\frac{(x-1960)}{10}\right)^3 - 45245\left(\frac{(x-1960)}{10}\right)^2 + 74567\left(\frac{(x-1960)}{10}\right) - 39214$$

Equation 1

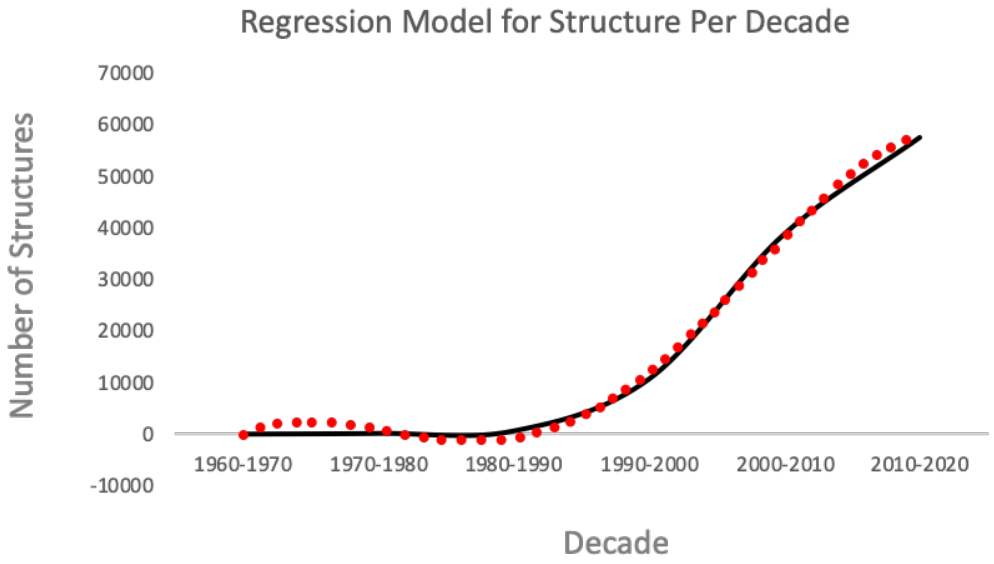


Figure 7- Graph of regression model compared to the actual data resulting from the declarative SQL output from the DIPSS database.

There were several other queries that were implemented that all centered around what information an end user may be interested in gathering. For example, the following are question guided queries that were asked: (1) What GO terms are found the most often for proteins in the database?; (2) How many GO molecular functions are the in the DIPSS database?; (3) How many proteins are related to apoptosis (cellular death)? Each of these queries answer incredibly fundamental questions easily with the use of the DIPSS database.

Code

There were several sophisticated functions implemented in the DIPSS database. There were six different functions implemented in the DIPSS database, four of which are modularized precursors to the primary prediction functions. These functions were implemented using imperative SQL- an important factor to consider when evaluating the DIPSS database dependencies. The DIPSS database only needs postgres SQL to function, which may be an important factor if installing the database in a closed container. The six functions are described in ***Table 2***. As illustrated in ***Figure 8***, the three major functions predict protein secondary structure, potential molecular functions, and potential cellular functions.

Table 2- A brief outline of the functions implemented in the DIPSS database.

Definitions for the Functions in DIPSS			
Function Name:	Input:	Output:	Description:
get_kmers	(1) Protein Seqence (2) Kmer Length	Kmers for the input Sequence	This function takes in a protein sequence and the kmer length, and returns the kmers for the input protein sequence.
findLikeProteins	(1) Protein Seqence (2) Kmer Length	Matching Kmer and the UniProt ID for the similar protein	This function takes in a protein sequence and the kmer length, and returns proteins with matching kmers and the UniProt IDs for those proteins
findSimiliarSecondaryStructures	(1) Protein Seqence (2) Kmer Length	List of secondary structures with the number of hits	This function takes in a protein sequence and the kmer length, and returns secondary structures corresponding to the simliar k-mers
predictSecondaryStructure	(1) Protein Seqence (2) Kmer Length	List of potential secondary structures and the rank	This function takes in a protein sequence and the kmer length, and returns secondary structures corresponding to the simliar k-mers
findPotentialCellularFunctions	(1) Protein Seqence (2) Kmer Length	List of potential cellular functions and the rank	This function takes in a protein sequence and the kmer length, and returns a ranked list of potential cellular functions for the input protein.
findPotentialMolecularFunctions	(1) Protein Seqence (2) Kmer Length	List of potential molecular functions and the rank	This function takes in a protein sequence and the kmer length, and returns a ranked list of potential mlecular functions for the input protein.

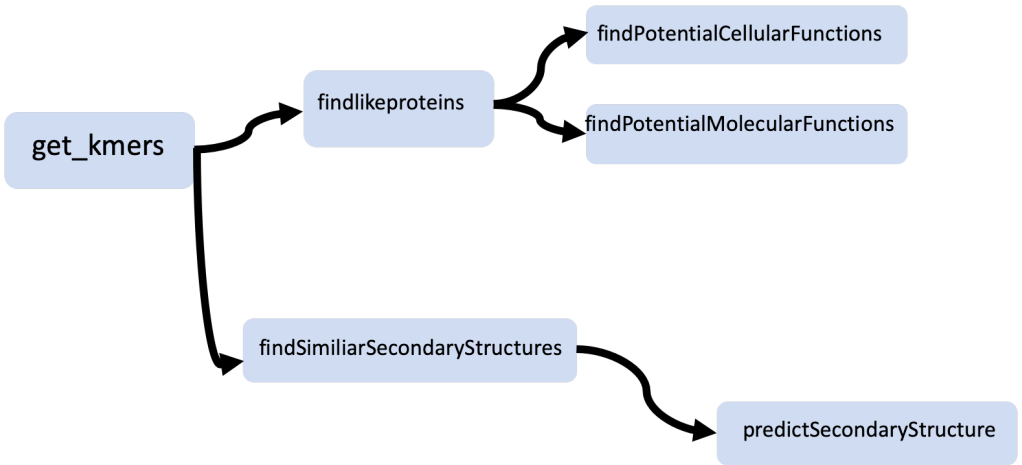


Figure 8- Overview of the imperative SQL functions, showing the flow of function input and output, allowing for a flexible modular design.

The function for predicting the secondary structure was implemented using a heuristic algorithm. It is believed that this algorithm is unique for predicting the secondary structure of proteins, however a further literature investigation would need to be conducted to be conclusive. The algorithm design was based around biochemical genetics, where both the sequential genetic sequence is known to play the primary role in dictating the resultant structure of the protein. In essence, the idea is that similar *known* sequences are more likely to produce the same secondary structures in other proteins. In other words, similar sequences give similar structures. This is based in the biological concept of structural "homology". The algorithm works by extending a string, with the string being the "H", "E", and "C" used in the secondary structure annotation.

This algorithm is carried out in a seven-step process, first taking in the protein sequence and the user-defined kmer size. Thereafter, the kmers are mapped to other proteins in the database, that have defined secondary structures. The objective is to find other proteins sharing subsequence kmers. Once these proteins have been identified, the secondary structure abbreviation (H, E, or C) matching the interval of overlap for the kmer is stored in a temporary table (***Figure 9, A***). This process is performed for each kmer in the input string. The next step labels each of these secondary structures according to which kmer they are associated with, so the secondary structures found with the first kmer of an input sequence are all given the index 1 (***Figure 9, B***). Thereafter a stitching process is performed, where the secondary structure kmers are elongated by one amino acid at a time, assuming there is sufficient overlap in the sequence (i.e. the end of the secondary structure sequence for index i overlaps the sequence of the secondary structure for index i+1) (***Figure 9, C***). If there is no overlap between the indexes, then elongation is not performed for that particular secondary structure abbreviation. Once this process is performed for each indexed secondary structure kmer, the final result can be thought of a directed acyclic graph, where the leaf nodes indicate the final elongated secondary structure, following the overlap sequences from root to leaf (***Figure 9, D***). However, this results in many potential secondary structures. To choose the most likely, the edges are weighted based on the number of times a particular secondary structure appears in the database for a particular protein sequence kmer (***Figure 9, E***). After weighting the edges, the highest weighted root-to-leaf path then is used to represent the most likely secondary structure for the protein (***Figure 9, F&G***).

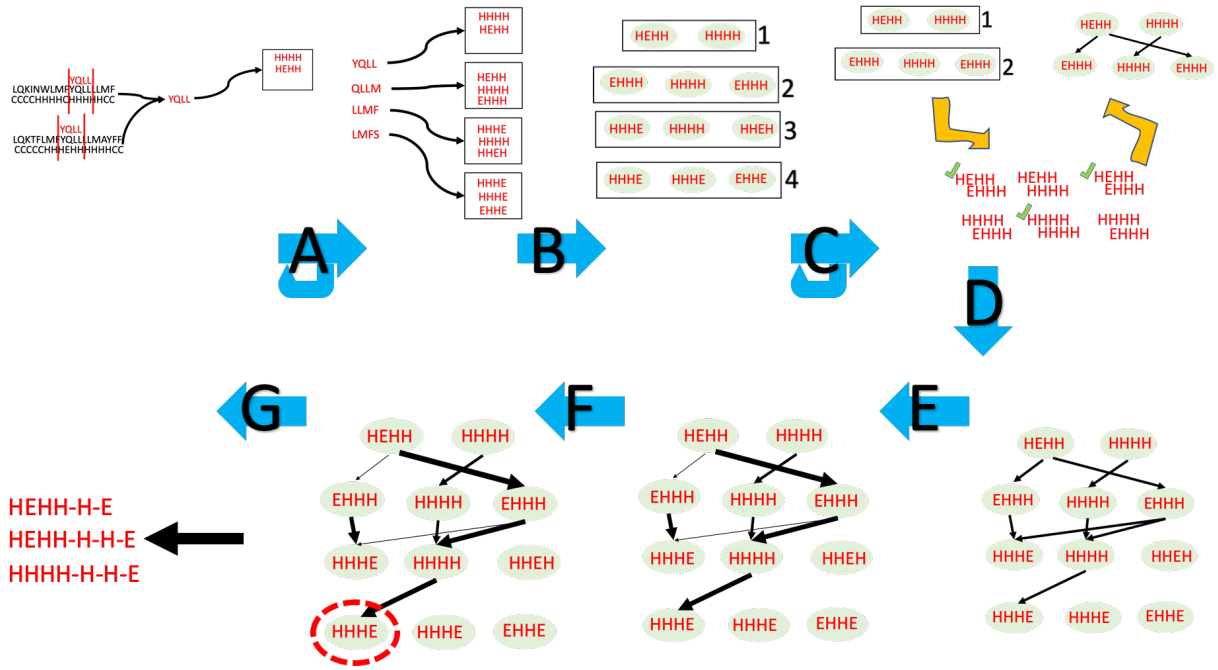


Figure 9- Algorithmic process for the secondaryStructurePrediction function implemented in the DIPSS database. The blue arrows indicate the steps involved in the process, and the arrows with circular inner arrows indicate iterative processes.

Files

There are two files that are associated with the DIPSS project. These are separated into two files containing the SQL code for the project. First is the SQL file containing the queries for the declarative SQL, while the second file contains information/code for the functions implemented in imperative SQL.

Table 3- Files supplied with the DIPSS project report

File Name:	File Description:
declarativeSQL.sql	This file contains declarative SQL queries based on potential user needs. These sample queries supply example inquires that an end user may have addressed by the DIPSS database.
imperativeSQL.sql	The imperativeSQL.sql file contains code that was used to build the functions for the DIPSS database. This includes the all of the functions used in their modular form.

Results

To test the efficacy of the functions within the DIPSS database, and simultaneously the database potential and usability, known structures from the protein database were used as case studies. These are *case study examples* showing the potential of the DIPSS database, but it is noted that a more comprehensive analysis would present potential end users with a statistically verified database.

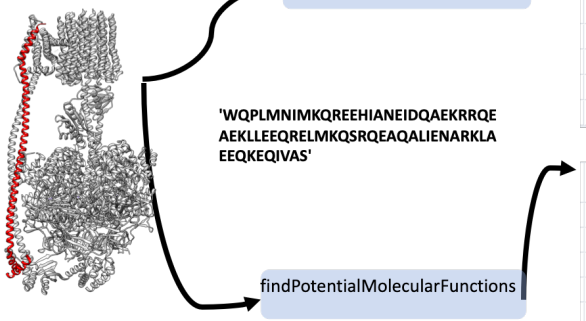
ATP Synthase Case Study

The first protein used to test the imperative SQL functions within the DIPSS database was ATP synthase. ATP synthase is the final protein encountered in the electron transport chain within mammalian cells, which acts to convert Adenosine Diphosphate (ADP) into an energetically useful derivative, Adenosine Triphosphate (ATP), using a proton gradient stored in the inner membrane of the mitochondria. Besides the biological importance of the protein, the interesting structure of the protein makes it a useful protein for evaluating the DIPSS database.

Because of the giant size of ATP synthase, only a portion of the protein was used to query the database. This is mainly due to the time complexity involved with the protein structure algorithm, since it is believed that finding the potential molecular and cellular functions should scale well with increasing length of the protein. To test a portion of ATP synthase, the long alpha helix ranging from end to end was queried. This sequence corresponds to "*WQPLMNIMKQREEHIANEIDQAEKRRQEAEKLLEEQRELMKQSRQEAQALIENARKLAEEQKEQI VAS*", which is a string of amino acids involved in the alpha helix in the structure.

Testing the ability of the functions within DIPSS to predict cellular and molecular functions of ATP synthase was performed using the alpha helix sequence. Using the tables within information from the Gene Ontology, the highest corresponding terms matching k-mers from the proteins sequence were ranked higher (***Figure 10***). This method of ranking based on the number of times a k-mer matched a substring worked incredibly well, showing the ability for kmer based methods to retrieve potential related GO terms with high accuracy. As for the molecular functions of ATP Synthase, terms such as "mitochondrial inner membrane", "integral component of the membrane", and "mitochondrial electron transport" were all among the top hits. The function for predicting cellular processes also worked well, showing "aerobic respiration", "oxidation-reduction process", and "mitochondrial electron transport" as all being terminology likely correlated with the input sequence. This shows the unique ability for the functional prediction functions to assist with the DIPSS queries.

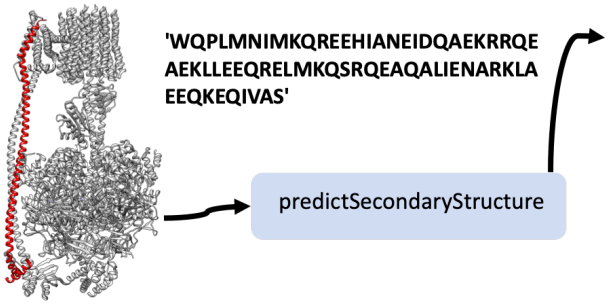
The next test was focused on the ability of DIPSS functions to predict the secondary structure of the ATP synthase alpha helix structure. Because the only portion used was entirely alpha helix structure, the correct structure prediction should be composed entirely of "H" characters. When used in the prediction function, the outcome was a string of all "H" characters. The output for the query was able to reach all the way to the bottom leaf node, extending the secondary structure prediction all the way to the full length of the protein. This result shows the ability for secondary structure prediction algorithm to work well on continuous secondary structure motifs, though it does not give a good estimate on the performance of the algorithm for proteins with more complex structures.



go_id character varying	molecularfunction character varying	hits bigint
'GO:001602'	'integral component of membrane'	18064
'GO:000574'	' mitochondrial inner membrane'	10726
'GO:007046'	' respirasome'	8313
'GO:000813'	' NADH dehydrogenase (ubiquinone) activity'	5939
'GO:000412'	' cytochrome-c oxidase activity'	4838
'GO:004687'	' metal ion binding'	4160
'GO:003196'	' mitochondrial membrane'	3689
'GO:000612'	'mitochondrial electron transport NADH to ubiquinone'	3024
'GO:002003'	' heme binding'	2716
'GO:000906'	' aerobic respiration'	2691

go_id character varying	cellularfunction character varying	hits bigint
'GO:000612'	'itochondrial electron transport NADH to ubiquinone'	3022
'GO:000906'	'aerobic respiration'	2691
'GO:000611'	' oxidative phosphorylation'	2689
'GO:004277'	'ATP synthesis coupled electron transport'	1979
'GO:005511'	'oxidation-reduction process'	1866
'GO:001598'	'ATP synthesis coupled proton transport'	1623
'GO:001964'	'aerobic electron transport chain'	1190
'GO:000612'	'itochondrial electron transport ubiquinol to cytochrome c'	931
'GO:003015'	'cell differentiation'	202
'GO:002290'	'respiratory electron transport chain'	174

Figure 10-Using ATP synthase as a protein to evaluate DIPSS. The top table shows the output for using the alpha helix structure, colored red, to query potential cellular functions. The bottom table shows the results of using the red alpha helix to find potential molecular functions.




```
--Predict secondary structure
SELECT *
FROM predictSecondaryStructure('WQPLMNIMKQREEHIANEIDQAEKRRQEAI
ORDER BY proteinlength DESC, weighttotal DESC;
```

	proteinsecondarysequence text	weighttotal bigint	proteinlength integer
1	HHHHHHHHHHHHHHHHHHHH...	29237	50
2	HHHHHHHHHHHHHHHHHHHH...	29146	50
3	HHHHHHHHHHHHHHHHHHHH...	29087	48
4	HHHHHHHHHHHHHHHHHHHH...	27551	48
5	HHHHHHHHHHHHHHHHHHHH...	27503	47
6	HHHHHHHHHHHHHHHHHHHH...	27409	46
7	HHHHHHHHHHHHHHHHHHHH...	27397	45
8	HHHHHHHHHHHHHHHHHHHH...	26860	44
9	HHHHHHHHHHHHHHHHHHHH...	26581	43
10	HHHHHHHHHHHHHHHHHHHH...	26553	43
11	HHHHHHHHHHHHHHHHHHHH...	26343	43

Figure 11- The ability of the secondary structure prediction used within the DIPSS database, validated using the alpha helix structure colored in red.

GFP Case Study

The case study using ATP synthase worked well to test the DIPSS functions on the alpha helix stretching across the ATP synthase structure, though it does not provide information on the efficacy of the structure prediction algorithm. To test the ability for DIPSS to work on

proteins with more intrinsically complicated structures, the goal was to test the ability of DIPSS to predict the secondary structure of Green Fluorescent Protein (GFP). This was an interesting protein to target, as many blind programs/algorithms may miss the random alpha helix structure in the program. In addition, the ability to engineer fluorescent proteins is of a high value, so this works as a validation case for the usefulness of DIPSS. The outcome was that the structure prediction was able to capture the basic (beta sheet-random coil) pattern seen in beta barrels proteins, but it was not able to fully recover the exact structure of the GFP secondary structure (***Figure 12***). Interestingly, the function was able to also predict areas between beta sheets that contained alpha helix structure (shown in red). It should also be noted that the function was not able to predict the full length of the structure, as the max length for the furthest leaf node was shorter than the input sequence. This part of the sequence is underlined and italicized in Figure 12. An overall evaluation is extremely promising, since this is the first version of the function and algorithm, and this is the raw output. Further improvements are only speculative, but one may imagine applying a machine learning model to clean up the output sequence, or using a hidden markov model to choose the most likely sequence out of the sequences returned (trained using the database).

$$(-(\text{BETA SHEET})-(\text{RANDOM COIL})-)^{n}$$

```
Q-MASKGEELFT(10)GVVPILVELD(20)GDVNGHKFSV(30)SGEGEGDATY(40)GKL
R-CCCCCCCCCC(10)CCCCEEEEEE(20)EEEEECCEEE(30)EEEEEEEEEH(40)HHC

Q-TLKFICT(50)TGKLPVPWPT(60)LVTTLXVQCF(70)SRYPDHMKQH(80)DFFKSA
R-EEEEEEE(50)ECCCCCCCCH(60)HHHCCCCHHH(70)HCCCCCEEEE(80)EEEEEE

Q-MPEG(90)YVQEMTISFK(100)DDGNYKTRAE(110)VKFEGDTLVN(120)RIELKG
R-ECCE(90)EEEEEEEEEE(100)CCCCCCCCCC(110)CCECCCCCCE(120)EEEEEE

Q-IDFK(130)EDGNILGHKL(140)EYNYNSHNVY(150)ITADKQKNGI(160)KANFK
R-CCCC(130)EEEEEEEEEE(140)EECCCCEEEE(150)EEEEEEEECC(160)CCCCC

Q-IRHNI(170)EDGSVQLADH(180)YQQNTPIGDG(190)PVLLPDNHYL(200)STQSA
R-CCCCE(170)EEEEEEEEEC(180)CCCCCCCEEE(190)EEEEEEEECC(200)CC

Q-LSKDP(210)NEKRDHMVLL(220)EFVTAAGI
```

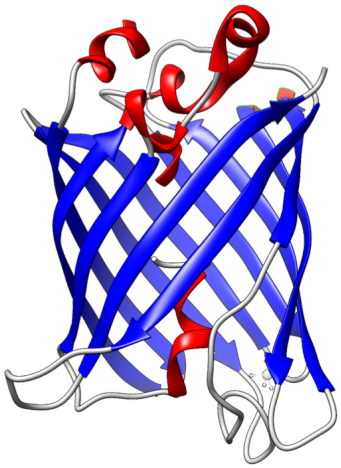


Figure 12- Secondary structure prediction outcome for the full length of GFP. The Q indicates the queried protein sequence and the R represent the returned predicted secondary structure for the input protein sequence.

Discussion

Design Decisions

The overall design of the database was changed throughout the building process. After considering multiple designs, the overall choice for DIPSS was made focusing on maximizing user accessible information. The goal was to have several databases incorporated, such as the Gene Ontology and UniProt, as well as the flexibility to incorporate new databases in future editions.

Challenges

Challenges constructing the DIPSS database were ubiquitous throughout the process of both building and testing the database. The first encountered was inability to use the database as desired because of the GO table being denormalization, and this resulted in difficulties encountered when normalizing the database. The algorithm formation took several hours, after deciding against regression and hidden markov models, and the implementation for the algorithm spanned a couple of days. In general, the imperative SQL was incredibly time consuming. The imperative SQL code for the database spanned 500 lines of code, and took a couple of weeks to fully implement.

Assumptions

The original assumptions for the database was a quick build with basic queries. The outcome of the database is far more complex and sophisticated than originally planned. In this respect, the DIPSS database far outpaced the original assumptions and expectations. Conversely, it was assumed that implementing the DIPSS database would be relatively simple. It was soon discovered that implementing the database, ensuring proper normal form, ensuring consistency, and thinking of worthy functions is an incredibly difficult process.

Changes from Original Plan

The original plan is far different than the outcome for the project. When starting the project, it was assumed I would be coupling the COMP533 project with research. For this reason, it was decided that I would work alone on the project. After starting to build the project, it was realized that coupling the research project with the database course was infeasible. After this decision, the plan to work on a different project was enacted, completely changing the scope of the project.

Future Directions-Next Steps

There are many different directions for the DIPSS database. First, the algorithm for predicting secondary structure is quite unique, so it may be worth further pursuit. In fact, there may be the ability to speed the algorithm using hash functions to find the proteins with similar

structure (minHash, ect.), and machine learning to select for the best resulting secondary structures. Also, a faster improvement for the DIPSS database secondary structure prediction function would be changing it into a greedy approach for speed improvements. This would allow for a cheaper time complexity. Lastly, another GO table focusing on the cellular location may be added, allowing for more information on the potential processes for the proteins.

Conclusion

The DIPSS database has the potential to assist potential users with curating protein information, sorting based on structural information, finding secondary structures, and predicting potential functions of input proteins. However, these are only among a subset of the potential use cases for the DIPSS database. As shown in this report, the DIPSS database not only has a rich collection of structural information, but also has unique functions allowing for unprecedented SQL based queries. These functions include the ability to predict molecular functions, cellular functions, and to predict the secondary structures of proteins. While the DIPSS database has multiple uses as it is currently implemented, the goal is to make further improvements to the database to test its fill ability in assisting potential users. The best case scenario would be online access for end users to query the database, allowing for the database to quickly garnish access. This would create a positive feedback for DIPSS improvements, as users would be able to report on difficulties or desired changes in real time.

References

Huang, P. S., Boyken, S. E., & Baker, D. (2016, September 14). The coming of age of de novo protein design. Nature. Nature Publishing Group. https://doi.org/10.1038/nature19946

Kabsch W, Sander C, Biopolymers. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. 1983 22 2577-2637. PMID: 6667333; UI: 84128824.

Stephen K Burley, Helen M. Berman, et al. RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy (2019) Nucleic Acids Research 47: D464–D474. doi: 10.1093/nar/gky1004.

The UniProt Consortium UniProt: a worldwide hub of protein knowledge Nucleic Acids Res. 47: D506-515 (2019)