BCB330 Final Presentation

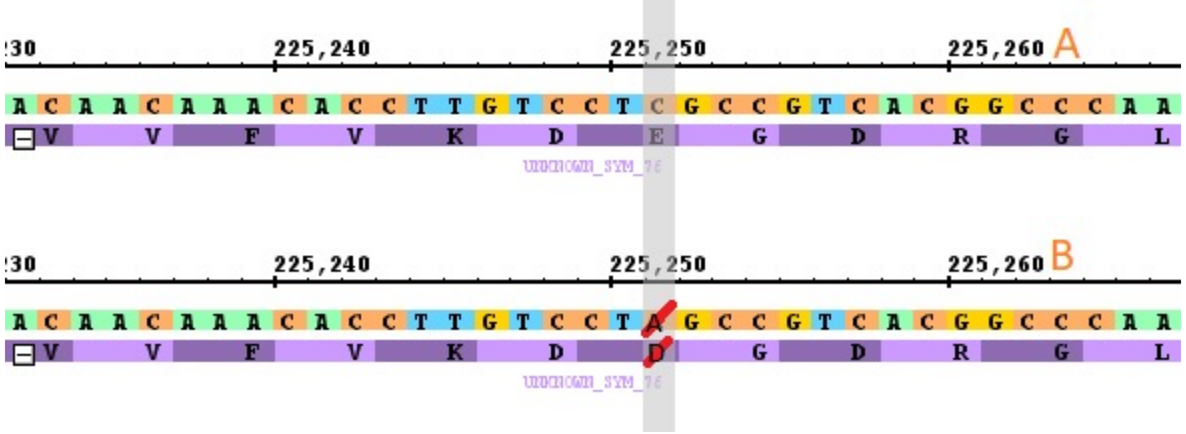
Extending multispecies non-synonymous SNP comparison and visualization with SNPer.

Glycine max. (L.) Merr.

- Common Name: Soy Bean
- Legume Plant from Fabaceae family
- Major Global Food Crop
  - Grown for Oil and Protein-Rich seeds
- Native to Asia
- Used as a model organism for some plant Species
- Researched for many uses including diabetes and aging treatment (Balode &Bodhankar, 2013)(Banerjee et al., 2021)

Single Nucleotide Polymorphisms (SNPs)

- Substitutions of Single Nucleotides within DNA sequences.
- Able to cause increased/decreased susceptibility to certain disease.
- May cause alterations in protein sequence/structure.
- Can be classified in 2 ways when found in DNA Coding Regions.
    - Synonymous – Does not results in a change in protein sequence.
    - Non-synonymous – Results in changes in the protein sequence.



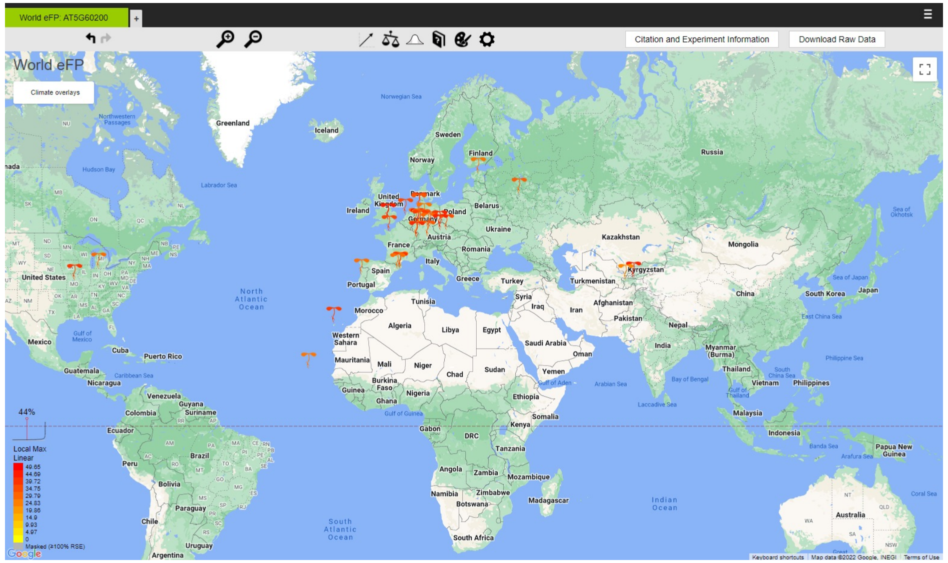
Example SNP (A: Before, B: After) from Glycine max chromosome 1 displayed using genome browser.
Nucleotide sequence is shown above translated, resultant protein sequence.

Variant Call Format(VCF) Files

- A tab-delimited flat-format text-file.
- Contains DNA variant information
- 1 variant per line (called “Record”)
- 1 + samples / variant

The BAR, ePlant and SNPer

- ePlant is a data visualization tool accessible from the Bio-Analytic Resource (BAR)
  - Allows visualization of plant information from the nanometer scale to kilometer scale.

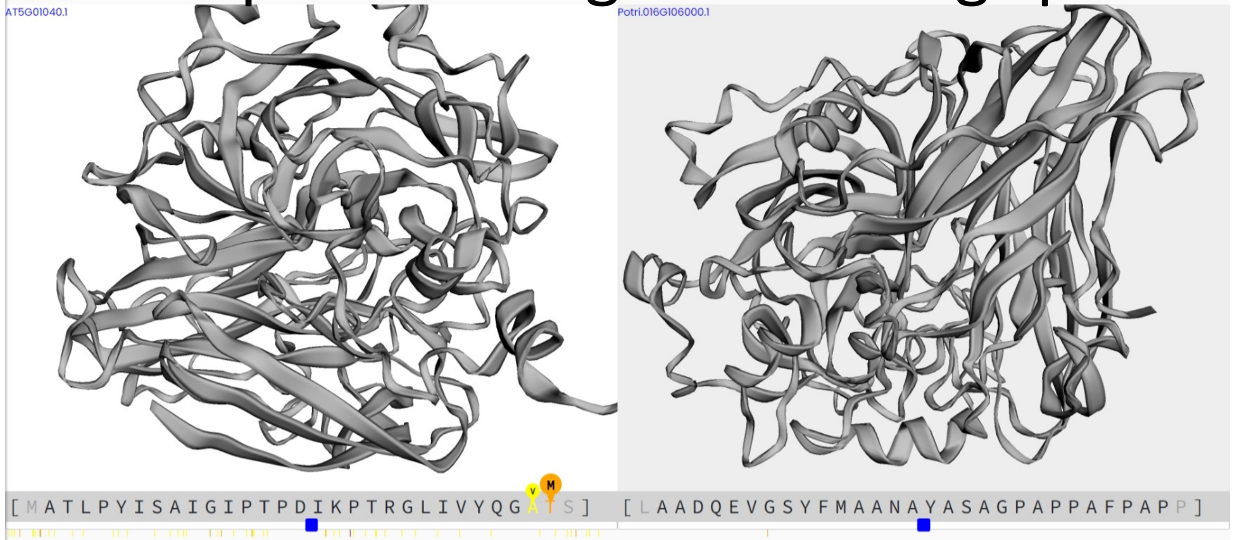


ePlant's World eFP tool, which shows expression data in a heat-map format for Arabidopsis Strains across the world.

ePlant's molecule viewer tool, which shows the corresponding protein molecule for the specified gene.

SNPer

- Existing tool SNPer as part of the BAR.
- Currently, SNPer has support for Arabidopsis, Poplar, and Tomato.
- Project seeks to database variants so Glycine max variant and protein information can be explored alongside existing species data.



Screenshot of the beta SNPer tool. SNPer was used to find an Arabidopsis gene (AT5G01040.1) and it's corresponding poplar homologue. (Potri.016G106000.1)

Project Overview

- Project split into 4 main components.
  1. Annotation of nsSNP data with SNPEff.
  2. Comparison of protein structure prediction tools.
  3. Structural prediction of *Glycine max* sequences for use with SNPer.
  4. Insertion of *Glycine max.* genetic variation data into the BAR ePlant database (for use with SNPer)

Materials: Data Sources

- SNP variation data provided by Dr. Anne Brown (USDA)
- 3 Datasets from soybase.org:
  - Valliyodan & Brown (2021) -- 481 accessions from the United Soybean Board (USB) Project
  - Torkamaneh & Laroche 2019 – Genetic Variation data from whole U.S. Soybean Germplasm collection.
  - Song & Hyten, 2015 – 1007 Accessions of Glycine max.

Materials: Software Tools

- Variant Annotation
  - SNPEff (5.0e) (Cingolani et. al. [GET YEAR])
- Structural Prediction
  - PHYRE2 (used for generation in SNPer)
  - AlphaFold2 (for comparison with PHYRE2)
  - PyMol (Version 2.4.0)
- Databasing
  - Python (Version 3.8.10)
  - SQL (Version 8.0 Community Edition)
- Quality Control
  - IGB (Version 11.1)

Annotation of nsSNP data with SNPEff

- SNPEff (5.0e) was used to annotate .vcf files containing genetic variation data.
- SnpSift (SNPEff tool)'s filter function was used to filter records which did not have a missense variant with moderate effect.

```
1       27721   A01.0000570     C       A       .       .       ANN=A|missense_variant|MODERATE|GLYMA_01G000100|GLYMA_01
G000100|transcript|KRH74114|protein_coding|3/3|c.250G>T|p.Val84Phe|352/718|250/315|84/104||     GT      0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
/0      0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0/0     0
```

Example record from TL2019 .vcf file after annotation.

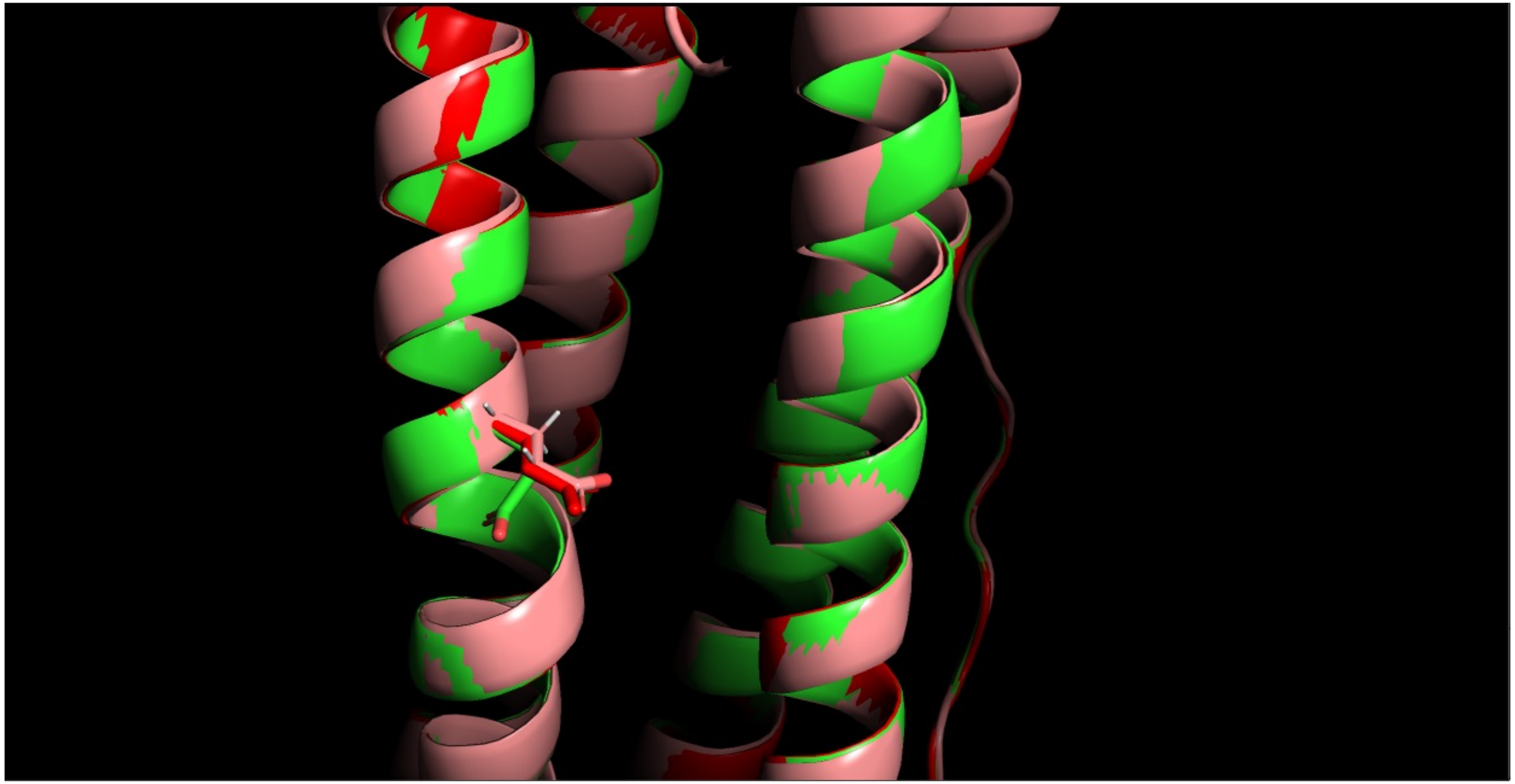
Structural Prediction Tool Comparison

- PHYRE2 compared with AlphaFold 2
  - Convenience/ ease of use.
  - Speed of algorithm.
  - Accuracy (RMSD of protein aligned with structure derived from X-ray crystallography)
- Soybean Ferritin SFER4 was used to compare (PDBs: wt=3a68, mut=3a9q)
  - Wild-Type and Mutant downloaded from pdb.
  - Predicted structures generated with each tool and aligned with PyMol.

Accuracy of Structural Prediction Tools

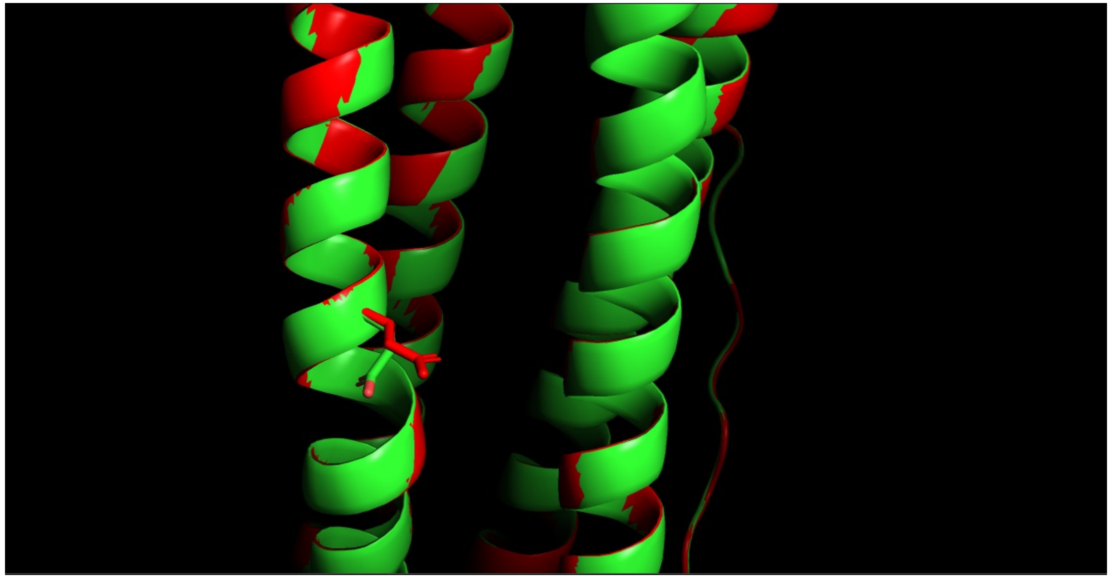
	RCSB Wild Type (PDB=3A68)	RCSB Mutant (PDB=3A9Q)	Phyre2 Predicted Wild Type	Phyre2 Predicted Mutant	AlphaFold2 Predicted Wild Type	AlphaFold2 Predicted Mutant
RCSB Wild Type (PDB=3A68)		0.156	0.168	0.168	0.400	0.207
RCSB Mutant (PDB=3A9Q)			0.154	0.154	0.388	0.202
Phyre2 Predicted Wild Type				0.000	0.522	0.233
Phyre2 Predicted Mutant					0.521	0.233
AlphaFold2 Predicted Wild Type						0.422
AlphaFold2 Predicted Mutant						

Comparing Wild-Type Structures

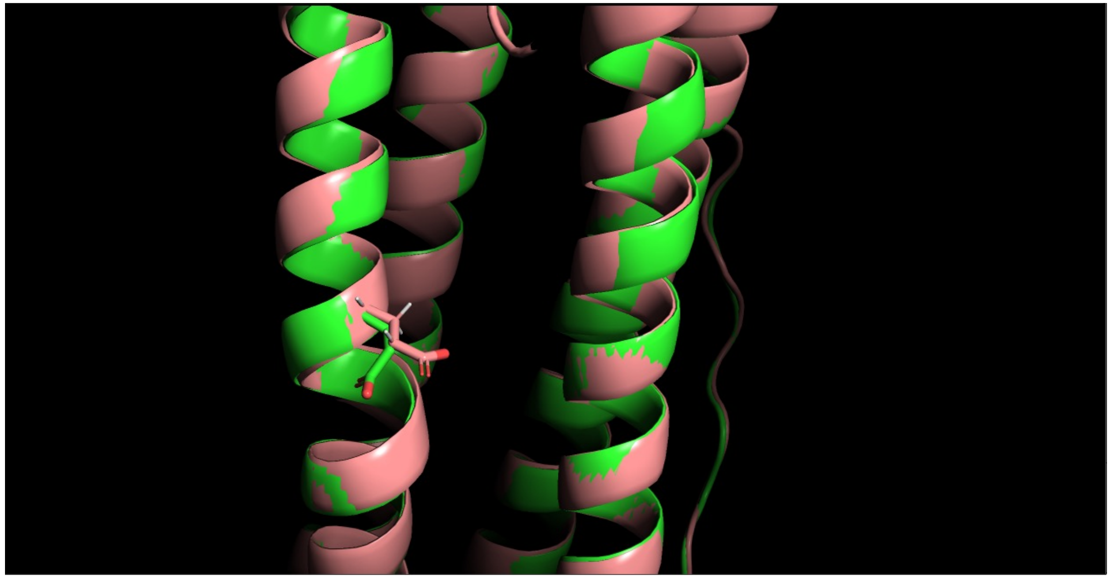


Phyre2 and AlphaFold predicted Wild-Type structures aligned with the canonical X-Ray structure (from RCSB.org)

Comparing Wild-Type Structures

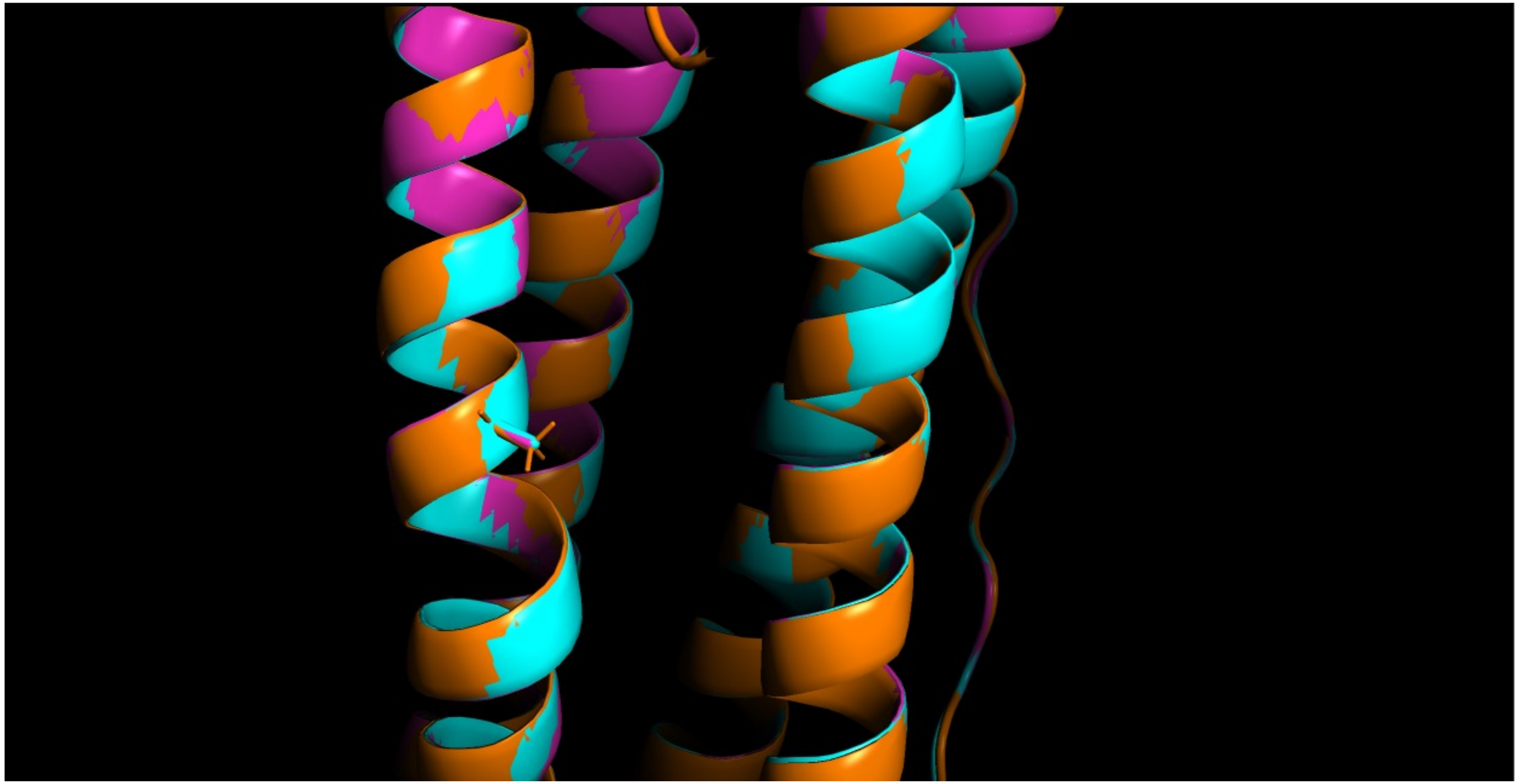


RCSB (X-Ray) vs Phyre2 (RMSD = 0.168)



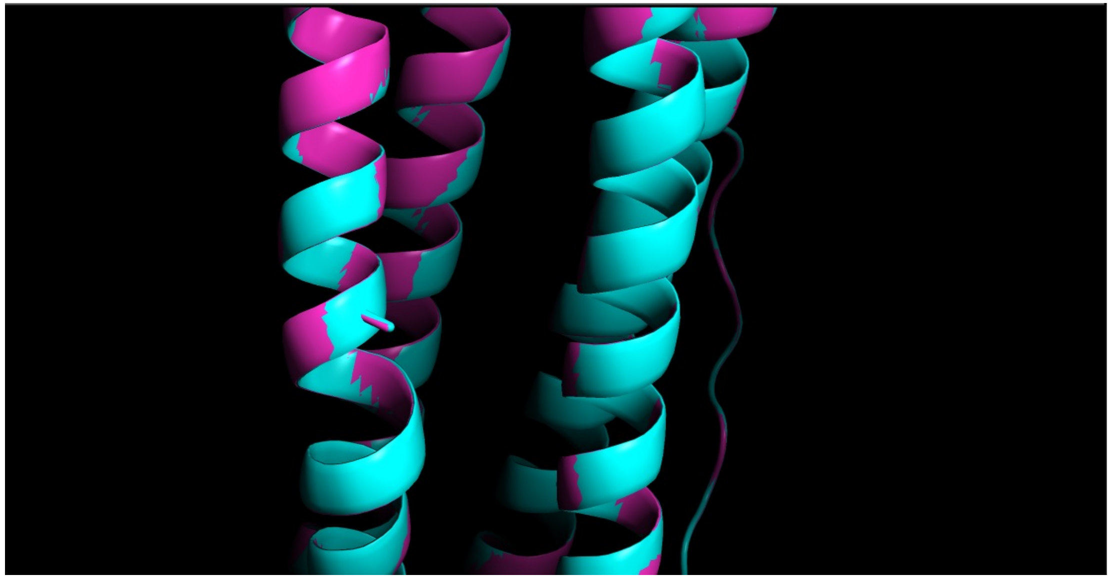
RCSB (X-Ray) vs AlphaFold2 (RMSD = 0.400)

Comparing Mutant Structures

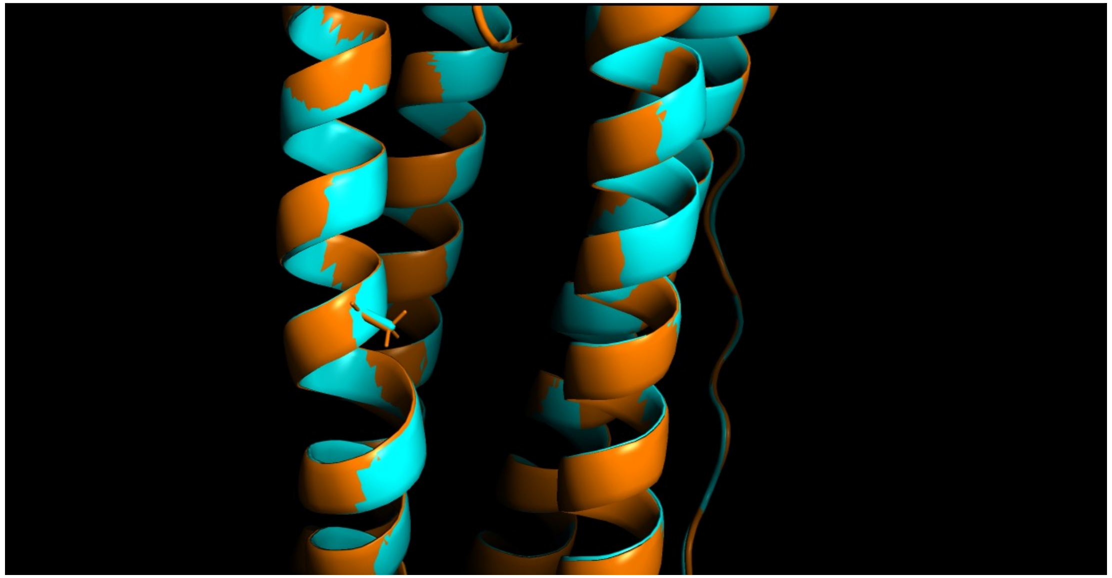


Phyre2 and AlphaFold predicted mutant (E173A) structures aligned with the canonical X-Ray structure (from RCSB.org)

Comparing Mutant Structures

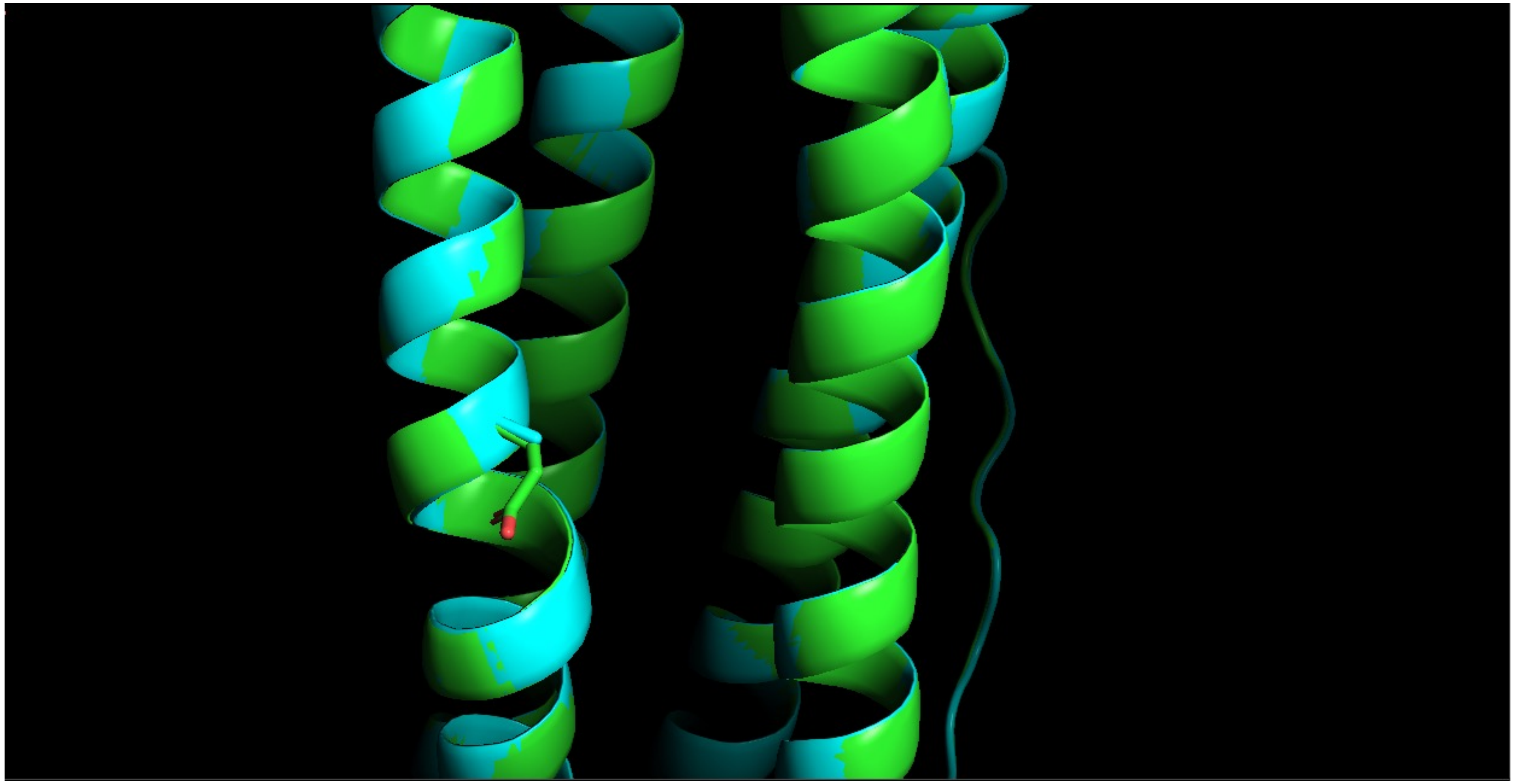


RCSB (X-Ray) vs Phyre2 (RMSD = 0.154)



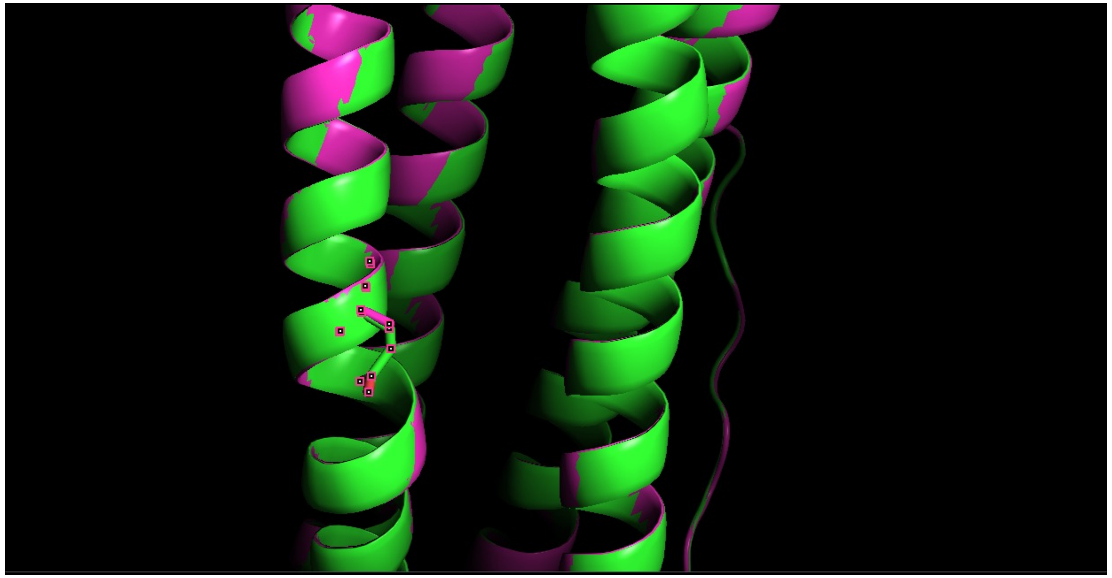
RCSB (X-Ray) vs AlphaFold2 (RMSD = 0.202)

Canonical Mutagenesis.

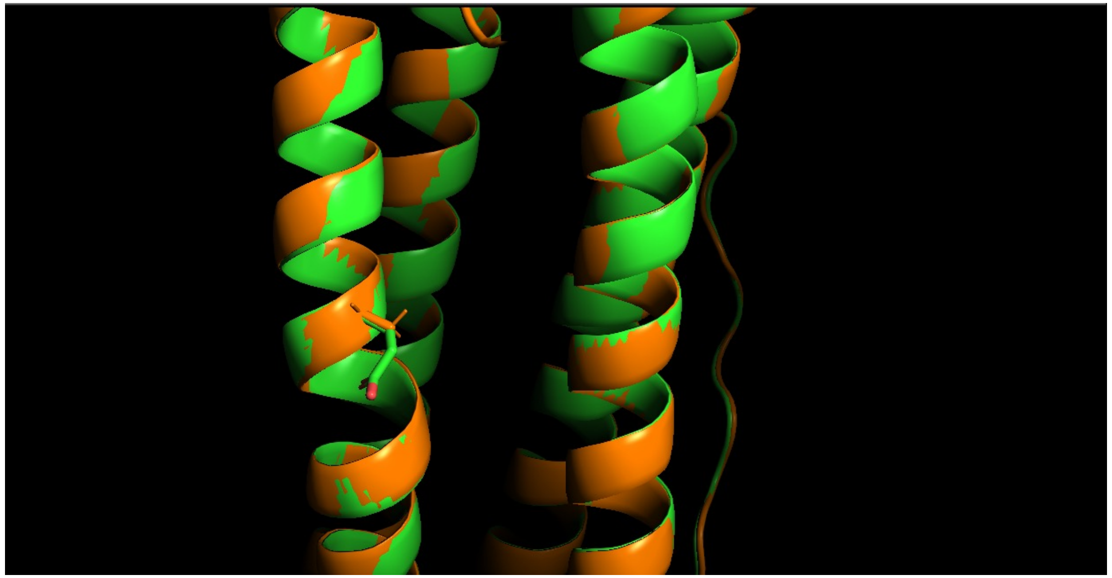


Wild-Type vs mutant (E173A) structures from RCSB.org aligned. (RMSD = 0.101)

Predicted Mutagenesis:



RCSB (X-Ray) vs Phyre2 (RMSD = 0.168)



RCSB (X-Ray) vs AlphaFold2 (RMSD = 0.207)

PHYRE2 vs AlphaFold

- As of testing, the current usability and accuracy of PHYRE2 make it a better option for the purposes of this project:
  - PHYRE2 is accessible via webform.
  - Sequences can also be submitted in a batch.
  - AlphaFold requires consistence use of Google Cloud GPUs to use, which has a toll on the speed and reliability of the results. (If GPU is not available)
  - AlphaFold is available for setup locally, however, this requires high computational power.

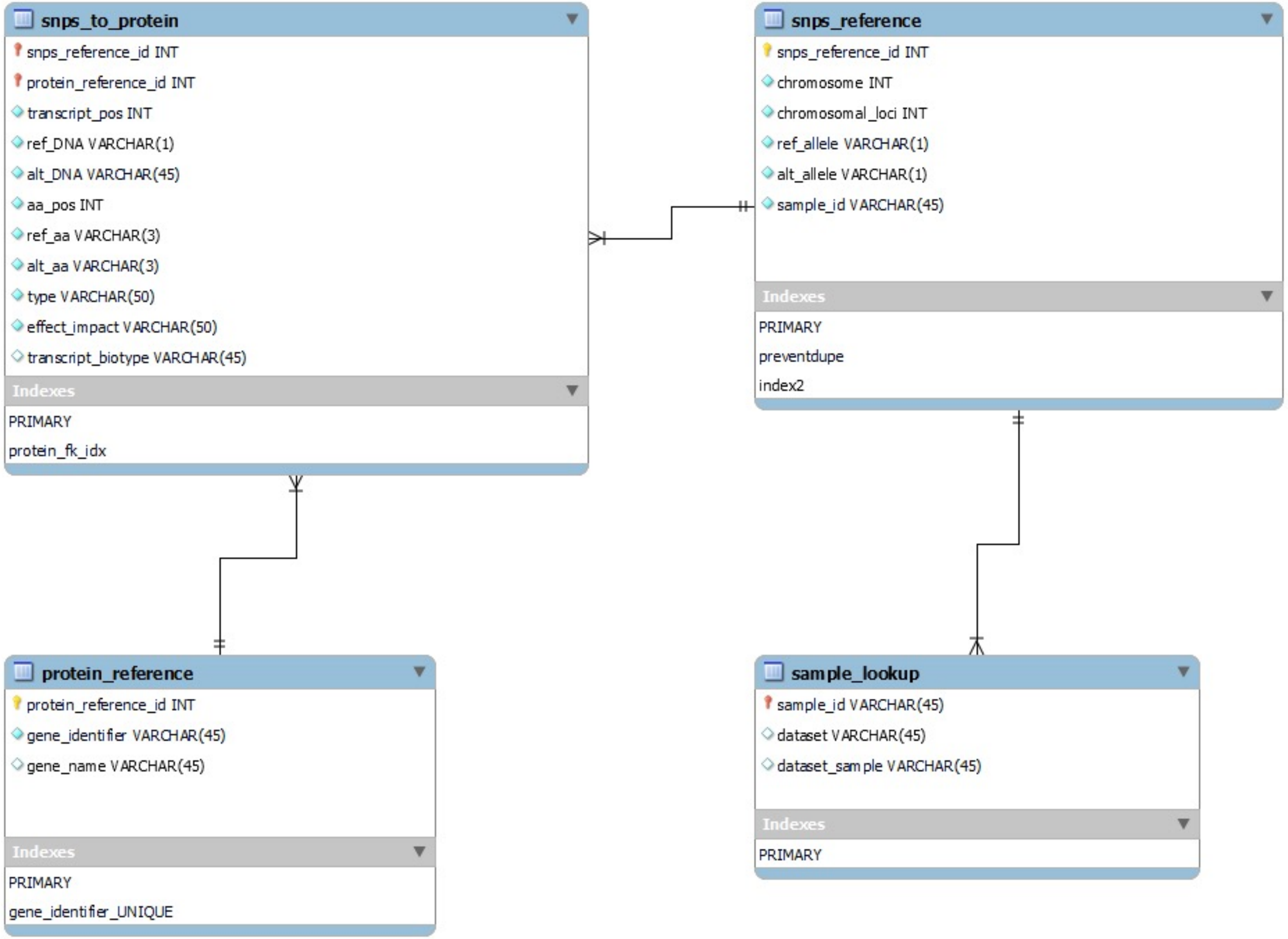
PHYRE2 Structural Prediction

- *Glycine max* 3-D Structures currently displayed by the SNPer tool have been previously generated with PHYRE2 and will be used for testing purposes within SNPer. (credit Kelsey Fuchs, Provart Lab)
- For testing, structures that are already available will be used for visualization.

Databasing Variants and their Effects.

- Databasing of Glycine max variants conducted on local machine.
  - Python and MySQL were used to parse the files and insert them into a database.
- After QC, database dump sent to Vincent for preparation of SNPer tool.

Database Schema



Python Script: Helper Functions

```
def connect():

    #Connect to database
    #:return: connection to db

    try:
        conn = MySQLdb.connect(host="localhost", user="terrence",
db="soybean_nssnp_local")
    except MySQLdb.Error:
        print("cannot connect to db")
        exit(1)
    return conn

def errorMsg(dbConn, file):
    dbConn.rollback()
    dbConn.close()
    file.close()
    exit(-1)
```

Database connection and error message reporting helper functions. (Credit: Vincent Lau)
`connect()` `errorMsg()`

Python Script: Main Function (1)

```
def main():

    try:
        annVCF = open(sys.argv[1], 'r')
    except FileNotFoundError:
        print("file cannot be found :(")
        exit(-1)

    conn = connect()
    cursor = conn.cursor()

    genoNames = []
    headerReached = False
    for line in annVCF:
        quit if headerReached else None
        if line[0] == "#" and line[1] == "#":
            #Skip over meta info lines:
            continue
        elif line[0] == "#":
            line = line.rstrip()
            columns = line.split('\t')
            genoNames = columns[10:]
            #INFO: print("GENOTYPEIDNAMES ARE: ", str(genoNames))
        else:
            #This is on ""regular"" lines (entry lines).
            line = line.rstrip()
            columns = line.split("\t")
            snpInfo = columns[0:10]
            genoTypeInfo = columns[10:]
            i = 0
            insertSnp = "INSERT INTO snps_reference (chromosome, chromosomal_loci, ref_allele, alt_allele, sample_id) VALUES 
(%s, %s, %s, %s, %s)"
            insertTranscriptId = "INSERT INTO protein_reference (gene_identifier, gene_name) VALUES (%s, %s)"
            insertSNPsToProtein = "INSERT INTO snps_to_protein \
                (snps_reference_id, protein_reference_id, transcript_pos, ref_DNA, alt_DNA, aa_pos, ref_aa, alt_aa, type, 
effect_impact, transcript_biotype) \
                    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
```

Main function pt. 1:

1. Opens the vcf file specified on the command line (argv)
2. Iterates through the header lines and collects the genotype information.
3. Collects record information per line
4. Prepares the SQL Queries we will be using to insert information into the database.

Python Script: Main Function (2)

```
for entry in genoTypeInfo:
  if "1" in entry[:3] and len(snpInfo[3]) < 2 and len(snpInfo[4]) < 2:
      try:
          cursor.execute(insertSnp ,(snpInfo[0], snpInfo[1], snpInfo[3], snpInfo[4], genoNames[i]))
      except MySQLdb.Error as e:
          print("Error inserting data into snp reference tbl " + line + " Error:" + str(e.args[0])
+ ", " + str(e.args[1]))
          errorMsg(conn, annVCF)
  else:
      continue

  snpIDPK = cursor.lastrowid
  print("snpIDPK is: ", snpIDPK)
  i=i+1

  infoList = columns[7].split("missense_variant")
  infoList.pop(0)
  transcriptIDPK = None
```

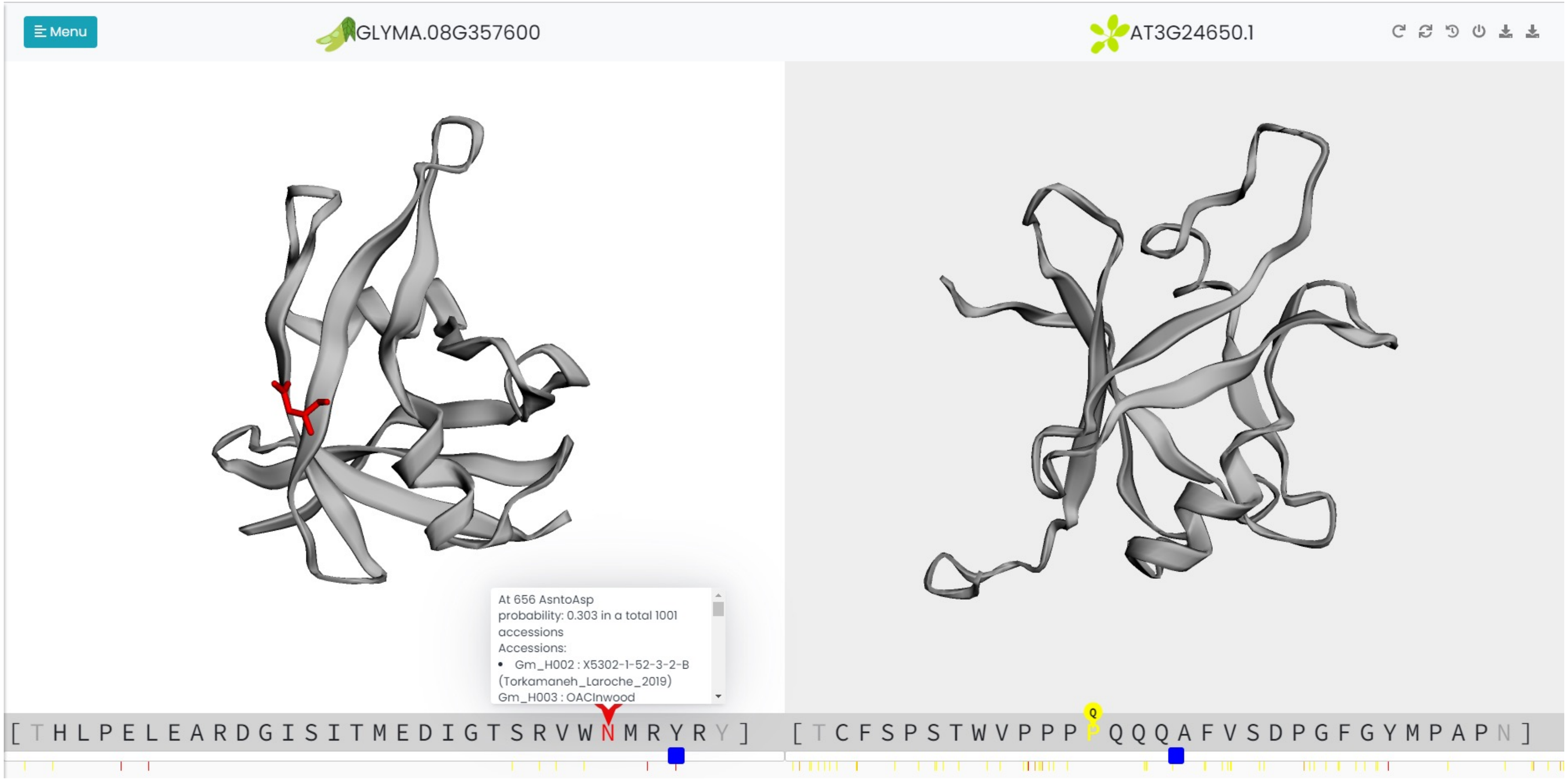
1. Iterates through the collected variant information and records variants with non zero genotype numbers (filters out records that contain != 1 reference or alternate alleles)
2. Inserts the variant information into the snps_reference table.
3. Retains snps_reference record id for use in the subsequent insertions.
4. Collects missense variant annotation information for use in subsequent insertions.

Python Script: Main Function (3)

```
for variant in infoList:
  missenseVarAnnoList = variant.split('|')
  if missenseVarAnnoList[1] != "MODERATE":
      print("non-MODERATE missense variant found!!!")
      continue
  transcriptId = missenseVarAnnoList[5].replace('transcript:', '')
  print(transcriptId)
  cursor.execute("SELECT * FROM protein_reference WHERE gene_identifier = %s LIMIT 1",
(transcriptId,))
  if cursor.rowcount == 0:
      try:
          cursor.execute(insertTranscriptId, (transcriptId, "",))
          transcriptIDPK = cursor.lastrowid
      except MySQLdb.Error as e:
          print("Error inserting into protein_reference table " + line + " Error:" + str(e.args[0])
+ ", " + str(e.args[1]))
          errorMsg(conn, annVCF)
  else:
      transcriptIDPK = cursor.fetchone()[0]
  print("(snpidpk, transcriptIDPK)", snpIDPK, transcriptIDPK)
  if transcriptIDPK is not None:
      try:
          cursor.execute("SELECT * FROM snps_to_protein WHERE snps_reference_id = %s AND
protein_reference_id = %s LIMIT 1", (snpIDPK, transcriptIDPK))
          if cursor.rowcount == 0:
              try:
                  cursor.execute(insertSNPsToProtein, (snpIDPK,transcriptIDPK, re.findall("\d+",
missenseVarAnnoList[8])[0], missenseVarAnnoList[8].split('>')[0][-1],
missenseVarAnnoList[8].split('>')[1][0], re.findall("\d+", missenseVarAnnoList[9])[0],
missenseVarAnnoList[9][2:5], missenseVarAnnoList[9][-3:], missenseVarAnnoList[4], "MODERATE", None
))
              except MySQLdb.Error as e:
                  print("Error inserting data into snp protein join tbl " + line + " Error:\n" +
str(e.args[0]) + ", " + str(e.args[1]))
                  errorMsg(conn, annVCF)
      except MySQLdb.Error as e:
          print("Error FINDING a snps_to_protein, (something might be really wrong)" )
          errorMsg(conn, annVCF)
conn.commit()
conn.close()
annVCF.close()
return 0
```

- Iterates through variants in a record and, for each:
  - Parses and collects variant annotation information
  - Filters out non-"MODERATE" SNP entries.
  - Checks for transcript-gene pairing in protein-reference table. (creates a transcript with unknown corresp. gene in case there is none).
  - Checks to see if sample has already been inserted in an earlier record, file or dataset.
  - Inserts SNP-Protein pairing information into snps_to_protein table.
  - Commits the changes to the database, closes connections to db and closes file.

SNPer Test Tool.



Conclusions and Future Outlook

- SNPer preliminary tests & quality control show:
  - Protein structures are generated successfully
  - SNPS have been annotated & filtered correctly.
  - Variation data inserted successfully into ePlant database.
- Features to be added to *Glycine max* subset of variation data in SNPer tool:
  - Protein structure view synchronization.