# Explanation of the steps of analysis & Result - Yuqi Huang

The PDB file of 17gs can be downloaded in https://www.rcsb.org/structure/17GS

**Complex Questions about Protein Structure:**

**Q1. How important is secondary structure in protein structure**

Need to know:

How the secondary structure types of proteins are defined

How to predict the secondary structure of a certain protein

How secondary structures contribute to the structural stability of proteins

Solution:

The secondary structures, alpha helices and beta sheets, are the building blocks for the tertiary structure of proteins. The arrangement and interaction of these elements largely determine the protein's three-dimensional shape. They are localized, regularly occurring structures in proteins and are primarily stabilized by hydrogen bonds. Alpha Helices (α-helices) are right-handed coils where each amino acid's backbone N-H group donates a hydrogen bond to the C=O group of the amino acid four residues earlier. They are characterized by 3.6 residues per turn. Beta Sheets (β-sheets) consist of beta strands that lie alongside one another, connected by hydrogen bonds. The strands can be parallel or antiparallel.

To predict the secondary structure of the protein of 17gs, we use DSSP program operated by Python. DSSP is a standardized algorithm for classifying amino acid residues in protein structures in secondary structure conformation, designed by Wolfgang Kabsch and Chris Sander.

DSSP can be installed successfully on ubuntu servers:

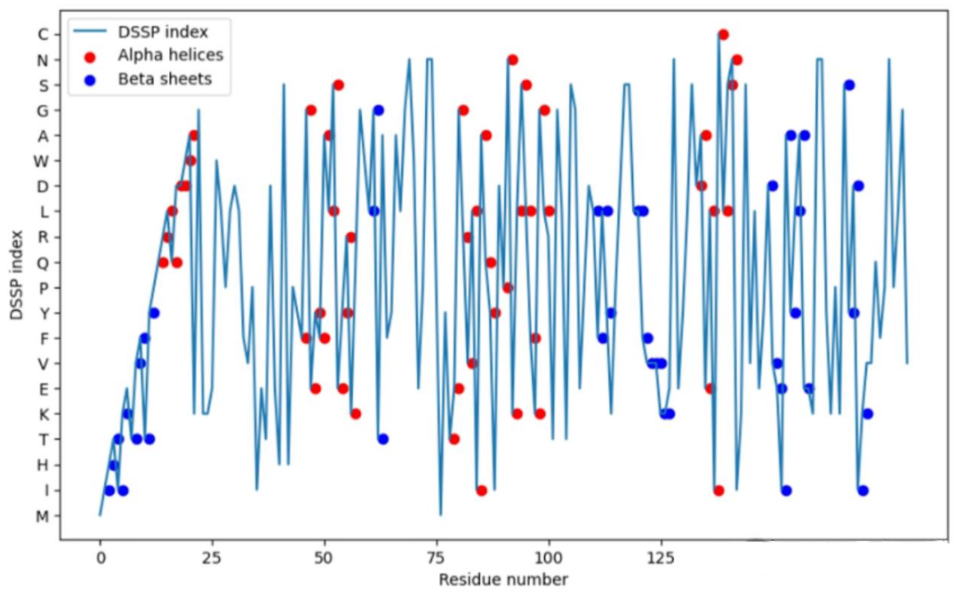
```
sudo apt-get install dssp
```

After importing the necessary packages, we started to analyze the protein files, and here we mainly implemented the sequence of fragments to print out the corresponding structure of the protein, as well as using the dssp metrics to draw graphs and do visualization. Here is the code:

```
from Bio.PDB import PDBParser
from Bio.PDB.DSSP import DSSP
import matplotlib.pyplot as plt

# Read PDB file
p = PDBParser()
structure = p.get_structure("protein_name", "~/17gs.pdb")

# Analyzing secondary structures using DSSP
model = structure[0]
dssp = DSSP(model, "~/17gs.pdb")
# Extracting fragments of alpha helices and beta folds
helices = [res for res in dssp if res[2] == 'H']  # Helix
sheets = [res for res in dssp if res[2] == 'E']  # Beta
```

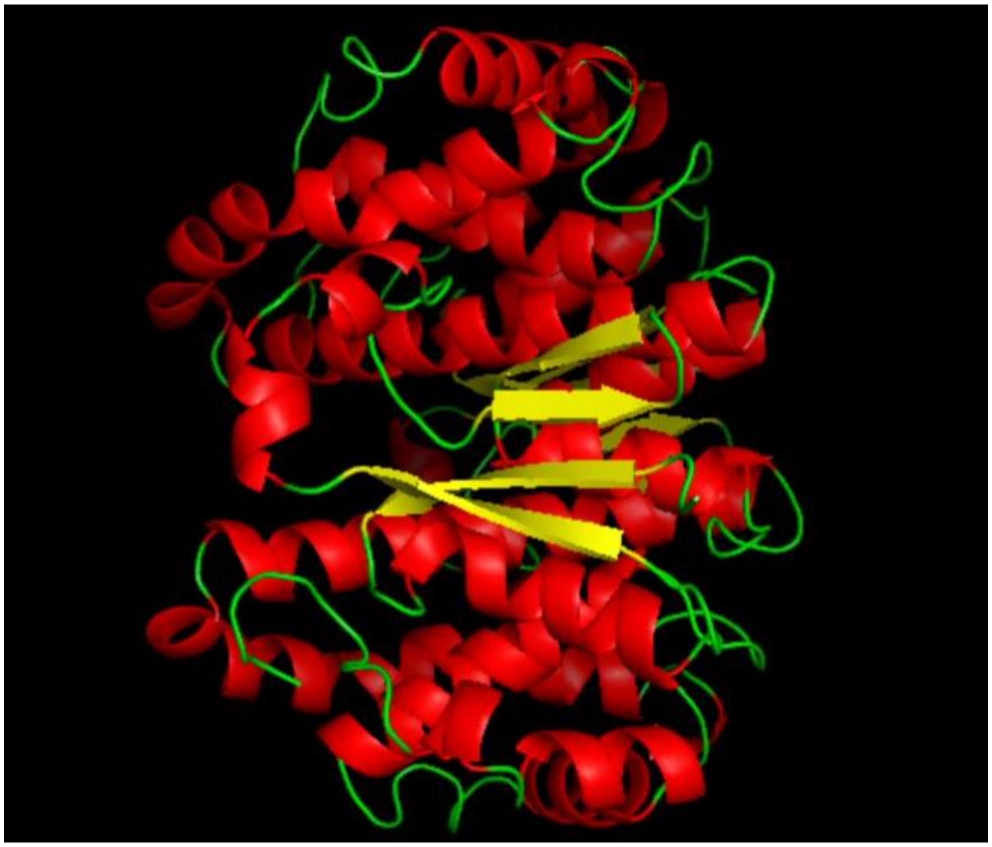
```
# Print segment information
print("Alpha helices:")
for res in helices:
    print(f"Residue {res[0]}: {res[1]}")

print("Beta sheets:")
for res in sheets:
    print(f"Residue {res[0]}: {res[1]}")

# visilization
plt.figure(figsize=(10, 6))
plt.plot([res[1] for res in dssp], label='DSSP index')
plt.scatter([res[0] for res in helices], [res[1] for res in helices],
color='r', label='Alpha helices')
plt.scatter([res[0] for res in sheets], [res[1] for res in sheets],
color='b', label='Beta sheets')
plt.xlabel('Residue number')
plt.ylabel('DSSP index')
plt.legend()
plt.savefig("01.png")
plt.show()
```

Then we get the result:



Alpha helices and beta sheets play a critical role in contributing to the overall structural stability of proteins. The alpha helix is a right-handed coil where every backbone N-H group donates a hydrogen bond to the backbone C=O group of the amino acid four residues earlier. This regular pattern of hydrogen bonds adds a significant amount of stability to the structure by reducing the entropy of the peptide chain and stabilizing the electrostatic interactions within the backbone. In beta sheets, the hydrogen bonds are formed between the backbone N-H groups of one strand and the C=O groups of the adjacent strand. These bonds can occur between strands that are parallel or antiparallel. The extensive hydrogen bonding network across multiple strands in beta sheets adds to the rigidity and flatness of the structure, providing a robust framework that resists unfolding. [1]

We can also use PyMol to find out the secondary structure of 17gs. 17GS is predominantly β-strands and α-helices. The β-strands are antiparallel and form a sheet-like structure.



[1] Stollar EJ, Smith DP. Uncovering protein structure. Essays Biochem. 2020 Oct 8;64(4):649-680. doi: 10.1042/EBC20190042. Erratum in: Essays Biochem. 2021 Jul 26;65(2):407. PMID: 32975287; PMCID: PMC7545034.

**Q2. Are there amino-acid composition variation between classes of protein folds?**

Need to Knows:

How can we classify and compare the amino acid compositions of different protein folds using bioinformatics databases?

What statistical methods or algorithms can be used to detect significant variations in amino acid composition between protein classes?

Can we employ machine learning techniques to predict protein fold class based on amino acid composition?

What evolutionary factors might explain the observed variations in amino acid composition between protein folds?

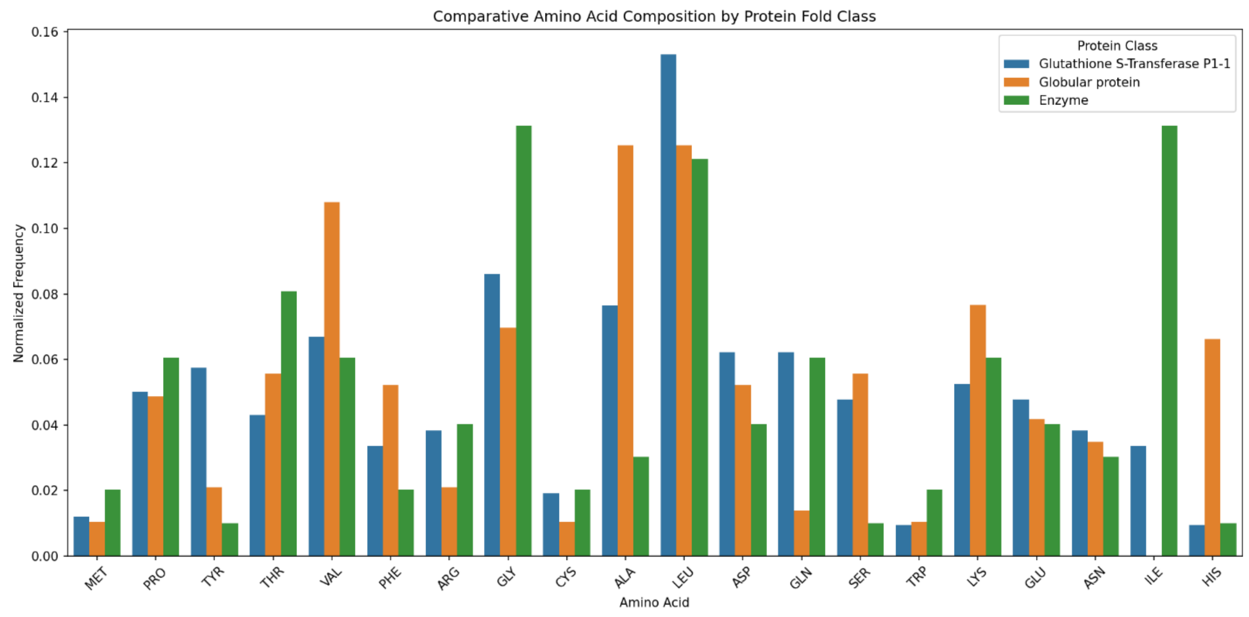
Solution:

Databases PDB provide extensive resources for studying different protein folds, classifying proteins not only by their amino acid sequences but also by their structural features. The we can use Biopython to parse protein structures from PDB. The amino acid compositions of these proteins can then be extracted and categorized based on their structural classification. Once they are composited, we can compare them across different classes by calculating the frequency of each amino acid within each class and visualizing these frequencies using histograms or pie charts to observe prominent trends. Here is the code:

```
import os
from Bio.PDB import PDBList, PDBParser
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

classifications = {
    '17gs': 'Glutathione S-Transferase P1-1',
    '4HHB': 'Globular protein',
    '1HSG': 'Enzyme'}


# Function to fetch PDB files and extract amino acid compositions
def fetch_and_analyze_structures(pdb_ids):
    pdbl = PDBList()
    parser = PDBParser(QUIET=True)
    aa_compositions = []

    for pdb_id in pdb_ids:
        file_path = pdbl.retrieve_pdb_file(pdb_id, pdir='.',
file_format='pdb', overwrite=False)
        structure = parser.get_structure(pdb_id, file_path)

        # Calculate amino acid composition
        aa_counts = {}
        for model in structure:
            for chain in model:
                for residue in chain.get_residues():
                    if residue.id[0] != ' ':
                        continue
                    res_name = residue.get_resname()
                    if res_name in aa_counts:
                        aa_counts[res_name] += 1
                    else:
                        aa_counts[res_name] = 1

        # Normalize counts to composition
        total = sum(aa_counts.values())
        composition = {aa: count / total for aa, count in aa_counts.items()}
        composition['PDB_ID'] = pdb_id
        composition['Class'] = classifications[pdb_id]  # Assign class
        aa_compositions.append(composition)
```

```python
    return aa_compositions


# List of PDB IDs to analyze
pdb_ids = ['17gs', '4HHB', '1HSG']

# Fetch data and calculate compositions
compositions = fetch_and_analyze_structures(pdb_ids)

# Convert to DataFrame for easier manipulation and visualization
df = pd.DataFrame(compositions).fillna(0)

# Melt the DataFrame for easier plotting
df_melted = df.melt(id_vars=['PDB_ID', 'Class'], var_name='Amino Acid',
value_name='Frequency')

# Plotting with seaborn
plt.figure(figsize=(14, 7))
sns.barplot(data=df_melted, x='Amino Acid', y='Frequency', hue='Class')
plt.title('Comparative Amino Acid Composition by Protein Fold Class')
plt.ylabel('Normalized Frequency')
plt.xlabel('Amino Acid')
plt.legend(title='Protein Class')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

The result turn out to be:



Here are some statistical methods to detect variations in amino acid composition:

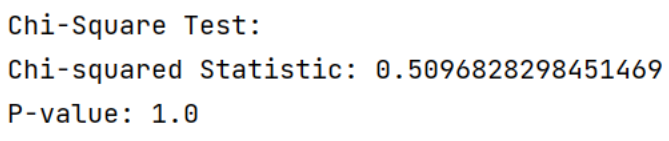
Chi-Square Test: This test can be applied to count data to compare the observed amino acid counts against expected counts derived from a larger reference set, or between two classes of protein folds.

ANOVA or Kruskal-Wallis Test: These tests can help determine if there are statistically significant differences in the proportions of specific amino acids across multiple classes of protein folds.

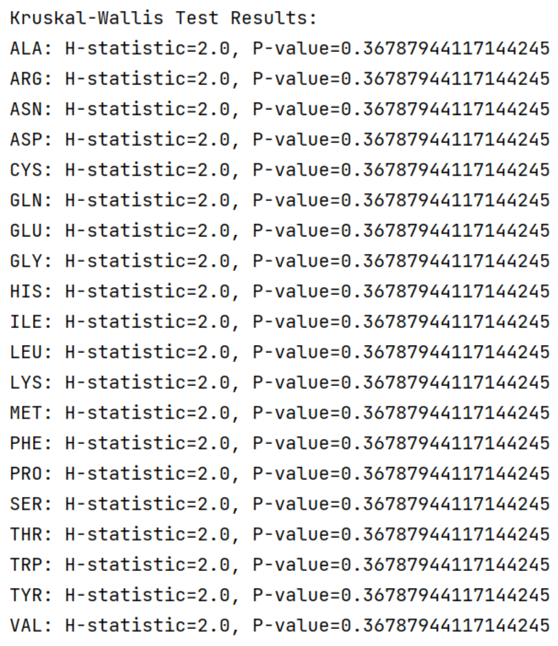
Here are the code for statistical:

```
# Prepare data for statistical testing
df_melted = df.melt(id_vars=['PDB_ID', 'Class'], var_name='Amino Acid',
value_name='Frequency')

# Chi-Square Test - aggregate data for chi-square test
contingency_table = pd.pivot_table(df_melted, values='Frequency',
index='Amino Acid', columns='Class', aggfunc=sum).fillna(0)
chi2, p, dof, expected = chi2_contingency(contingency_table.values)
print("Chi-Square Test:")
print("Chi-squared Statistic:", chi2)
print("P-value:", p)

# Kruskal-Wallis Test - apply test for each amino acid
print("\nKruskal-Wallis Test Results:")
for amino_acid in contingency_table.index:
    samples = [group.dropna().values for name, group in
df_melted[df_melted['Amino Acid'] ==
amino_acid].groupby('Class')['Frequency']]
    stat, p_value = kruskal(*samples)
    print(f"{amino_acid}: H-statistic={stat}, P-value={p_value}")
```

Then we can get the result:



```
Chi-Square Test:
Chi-squared Statistic: 0.5096828298451469
P-value: 1.0
```



```
Kruskal-Wallis Test Results:
ALA: H-statistic=2.0, P-value=0.36787944117144245
ARG: H-statistic=2.0, P-value=0.36787944117144245
ASN: H-statistic=2.0, P-value=0.36787944117144245
ASP: H-statistic=2.0, P-value=0.36787944117144245
CYS: H-statistic=2.0, P-value=0.36787944117144245
GLN: H-statistic=2.0, P-value=0.36787944117144245
GLU: H-statistic=2.0, P-value=0.36787944117144245
GLY: H-statistic=2.0, P-value=0.36787944117144245
HIS: H-statistic=2.0, P-value=0.36787944117144245
ILE: H-statistic=2.0, P-value=0.36787944117144245
LEU: H-statistic=2.0, P-value=0.36787944117144245
LYS: H-statistic=2.0, P-value=0.36787944117144245
MET: H-statistic=2.0, P-value=0.36787944117144245
PHE: H-statistic=2.0, P-value=0.36787944117144245
PRO: H-statistic=2.0, P-value=0.36787944117144245
SER: H-statistic=2.0, P-value=0.36787944117144245
THR: H-statistic=2.0, P-value=0.36787944117144245
TRP: H-statistic=2.0, P-value=0.36787944117144245
TYR: H-statistic=2.0, P-value=0.36787944117144245
VAL: H-statistic=2.0, P-value=0.36787944117144245
```

We can then use ML to predict protein fold class based on amino acid composition:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

# Preparing the data for machine learning
X = df.drop(['PDB_ID', 'Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Random Forest Classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Predictions
y_pred = clf.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
The result shows here:
 Classification Report:
                                 precision    recall  f1-score   support

                Globular protein       0.00      0.00      0.00       0.0
 Glutathione S-Transferase P1-1       0.00      0.00      0.00       1.0

                        accuracy                           0.00       1.0
                       macro avg       0.00      0.00      0.00       1.0
                    weighted avg       0.00      0.00      0.00       1.0

 Confusion Matrix:
  [[0 0]
  [1 0]]
```

**Complex Questions about Protein Function:**

**Q3. How can we predict the impact of protein-protein interactions on cellular signaling pathways?**

Need to Knows:

What bioinformatics resources are available to identify known protein-protein interactions relevant to a particular signaling pathway?

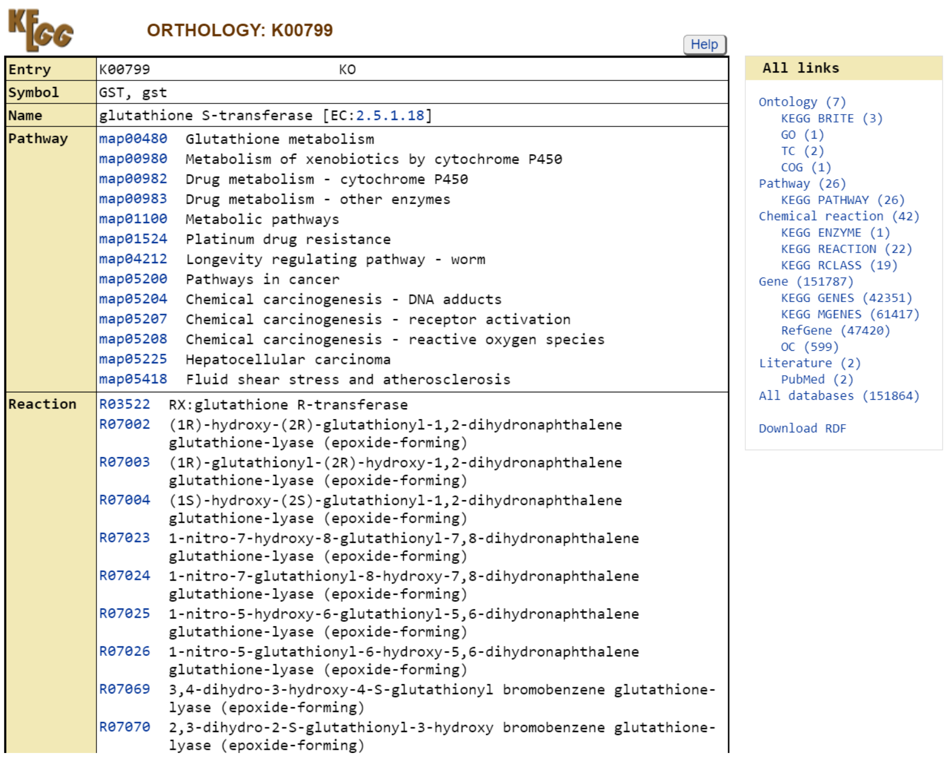
How can we simulate the dynamic aspects of protein-protein interactions affecting signal transduction using systems biology models?

What computational methods can be used to predict unknown protein-protein interactions based on structural complementarity and surface properties?

How do changes in protein conformation influence their interaction affinity and specificity within signaling pathways?

Solution:

To predict the impact of protein-protein interactions on cellular signaling pathways, we can firstly use databases to identify known protein-protein interactions relevant to particular signaling pathways. For example, KEGG can help understand high-level functions and utilities of the biological system from molecular-level information, especially large-scale molecular datasets generated by high-throughput experimental technologies. When we search about GST(The protein of 17gs), we get the following result:



ORTHOLOGY: K00799

Help

| | |
|---|---|
| Entry | K00799 KO |
| Symbol | GST, gst |
| Name | glutathione S-transferase [EC:2.5.1.18] |
| Pathway | map00480 Glutathione metabolism<br>map00980 Metabolism of xenobiotics by cytochrome P450<br>map00982 Drug metabolism - cytochrome P450<br>map00983 Drug metabolism - other enzymes<br>map01100 Metabolic pathways<br>map01524 Platinum drug resistance<br>map04212 Longevity regulating pathway - worm<br>map05200 Pathways in cancer<br>map05204 Chemical carcinogenesis - DNA adducts<br>map05207 Chemical carcinogenesis - receptor activation<br>map05208 Chemical carcinogenesis - reactive oxygen species<br>map05225 Hepatocellular carcinoma<br>map05418 Fluid shear stress and atherosclerosis |
| Reaction | R03522 RX:glutathione R-transferase<br>R07002 (1R)-hydroxy-(2R)-glutathionyl-1,2-dihydronaphthalene glutathione-lyase (epoxide-forming)<br>R07003 (1R)-glutathionyl-(2R)-hydroxy-1,2-dihydronaphthalene glutathione-lyase (epoxide-forming)<br>R07004 (1S)-hydroxy-(2S)-glutathionyl-1,2-dihydronaphthalene glutathione-lyase (epoxide-forming)<br>R07023 1-nitro-7-hydroxy-8-glutathionyl-7,8-dihydronaphthalene glutathione-lyase (epoxide-forming)<br>R07024 1-nitro-7-glutathionyl-8-hydroxy-7,8-dihydronaphthalene glutathione-lyase (epoxide-forming)<br>R07025 1-nitro-5-hydroxy-6-glutathionyl-5,6-dihydronaphthalene glutathione-lyase (epoxide-forming)<br>R07026 1-nitro-5-glutathionyl-6-hydroxy-5,6-dihydronaphthalene glutathione-lyase (epoxide-forming)<br>R07069 3,4-dihydro-3-hydroxy-4-S-glutathionyl bromobenzene glutathione-lyase (epoxide-forming)<br>R07070 2,3-dihydro-2-S-glutathionyl-3-hydroxy bromobenzene glutathione-lyase (epoxide-forming) |

All links

- Ontology (7)
  - KEGG BRITE (3)
  - GO (1)
  - TC (2)
  - COG (1)
- Pathway (26)
  - KEGG PATHWAY (26)
- Chemical reaction (42)
  - KEGG ENZYME (1)
  - KEGG REACTION (22)
  - KEGG RCLASS (19)
- Gene (151787)
  - KEGG GENES (42351)
  - KEGG MGENES (61417)
  - RefGene (47420)
  - OC (599)
- Literature (2)
  - PubMed (2)
- All databases (151864)

Download RDF

Then we can choose to simulate dynamic aspects of PPIs in signal transduction by using systems biology. For example, agent-based models and monte carlo simulations are useful for simulating the stochastic nature of molecular interactions in a spatially structured environment. Flux balance analysis can be adapted to explore the flow and regulation of signaling pathways.

To predict unknown PPIs based on structural complementarity and surface properties, we can use software like AutoDock and HADDOCK to predict how proteins might interact based on their structures by exploring how they fit together like pieces of a 3D puzzle. Also, ML is another way to get the result. Support vector machines or neural networks can predict PPIs based on patterns and features derived from known interactions.

We use python to firstly extract the information in pdb file:

```
from Bio.PDB import PDBParser
import MDAnalysis as mda
from MDAnalysis.analysis import align
import numpy as np

def parse_pdb(pdb_filename):
    parser = PDBParser()
    structure = parser.get_structure('PDB_structure', pdb_filename)
    for model in structure:
        for chain in model:
            print(f"Chain {chain.id}")
            for residue in chain:
                print(f"Residue {residue.get_resname()} at position 
{residue.get_id()[1]}")
```

To explore the impact of protein conformation on interaction affinity and specificity, we should do dynamic simulation. Molecular dynamics simulations usually require significant computational resources and expertise in setting up the simulations such as defining force fields, solvation models. (We can also use GROMACS to do dynamic simulation, the tutorial link is here: https://mybinder.org/v2/gl/gromacs%2Fonline-tutorials%2Fmd-intro-tutorial/main?filepath=tutorial.ipynb)

```
u = mda.Universe('17gs.pdb')

# Basic operations : RMSD calculation
rmsd = align.alignto(u, reference, select="backbone")
```

To analyze conformation changes over time or between different states, you might simulate or compare different structures:

```
def calculate_rmsd(structure_1, structure_2):
    R = align.rotation_matrix(structure_1.positions, structure_2.positions)
    rmsd_value = np.sqrt(np.mean(np.square(structure_1.positions - 
structure_2.positions.dot(R[0]))))
    return rmsd_value
```

Q4. How can we predict mutational hotspots within proteins that are critical for stability or activity?

Need to Knows:

What computational tools can identify conserved regions within a protein sequence that are likely to be sensitive to mutations?

How can molecular dynamics simulations contribute to understanding the effects of mutations on protein stability or function?

Can we use evolutionary conservation data to predict which mutations are likely to be deleterious?

What are the common features of known mutational hotspots that can be used to develop predictive models?

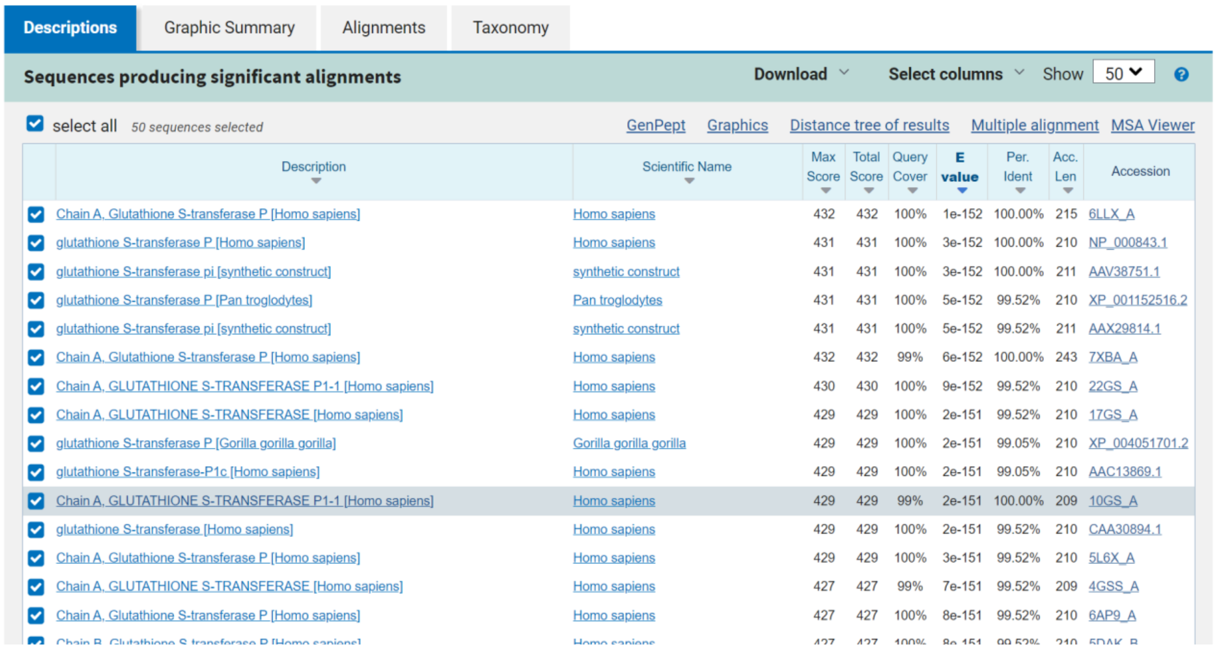
Solution:

To identifying conserved regions within protein sequences, we can follow these steps:

(1) Use BLAST in NCBI to get homologous sequences which are similar to 17gs protein sequence.

The protein sequence data for 17gs(GSTP1):
https://www.ncbi.nlm.nih.gov/protein/NP_000843.1?report=fasta

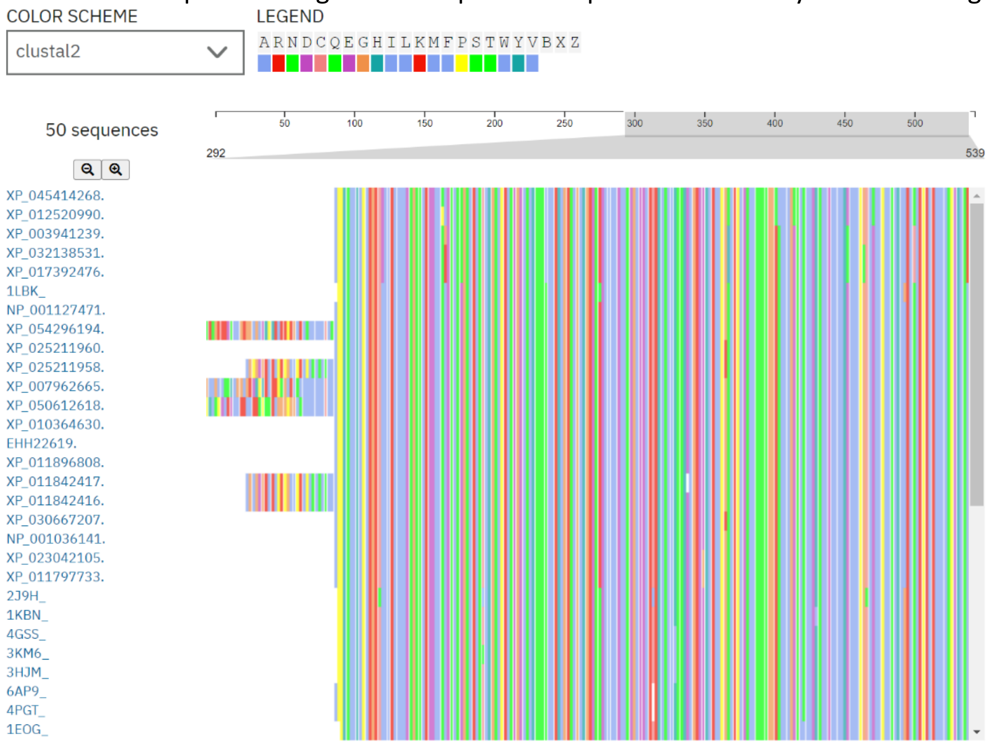
The result of BLAST:



Sequences producing significant alignments

| | Description | Scientific Name | Max Score | Total Score | Query Cover | E value | Per. Ident | Acc. Len | Accession |
|---|---|---|---|---|---|---|---|---|---|
| ✓ | Chain A, Glutathione S-transferase P [Homo sapiens] | Homo sapiens | 432 | 432 | 100% | 1e-152 | 100.00% | 215 | 6LLX_A |
| ✓ | glutathione S-transferase P [Homo sapiens] | Homo sapiens | 431 | 431 | 100% | 3e-152 | 100.00% | 210 | NP_000843.1 |
| ✓ | glutathione S-transferase pi [synthetic construct] | synthetic construct | 431 | 431 | 100% | 3e-152 | 100.00% | 211 | AAV38751.1 |
| ✓ | glutathione S-transferase P [Pan troglodytes] | Pan troglodytes | 431 | 431 | 100% | 5e-152 | 99.52% | 210 | XP_001152516.2 |
| ✓ | glutathione S-transferase pi [synthetic construct] | synthetic construct | 431 | 431 | 100% | 5e-152 | 99.52% | 211 | AAX29814.1 |
| ✓ | Chain A, Glutathione S-transferase P [Homo sapiens] | Homo sapiens | 432 | 432 | 99% | 6e-152 | 100.00% | 243 | 7XBA_A |
| ✓ | Chain A, GLUTATHIONE S-TRANSFERASE P1-1 [Homo sapiens] | Homo sapiens | 430 | 430 | 100% | 9e-152 | 99.52% | 210 | 22GS_A |
| ✓ | Chain A, GLUTATHIONE S-TRANSFERASE [Homo sapiens] | Homo sapiens | 429 | 429 | 100% | 2e-151 | 99.52% | 210 | 17GS_A |
| ✓ | glutathione S-transferase P [Gorilla gorilla gorilla] | Gorilla gorilla gorilla | 429 | 429 | 100% | 2e-151 | 99.05% | 210 | XP_004051701.2 |
| ✓ | glutathione S-transferase-P1c [Homo sapiens] | Homo sapiens | 429 | 429 | 100% | 2e-151 | 99.05% | 210 | AAC13869.1 |
| ✓ | Chain A, GLUTATHIONE S-TRANSFERASE P1-1 [Homo sapiens] | Homo sapiens | 429 | 429 | 99% | 2e-151 | 100.00% | 209 | 10GS_A |
| ✓ | glutathione S-transferase [Homo sapiens] | Homo sapiens | 429 | 429 | 100% | 2e-151 | 99.52% | 210 | CAA30894.1 |
| ✓ | Chain A, Glutathione S-transferase P [Homo sapiens] | Homo sapiens | 429 | 429 | 100% | 3e-151 | 99.52% | 210 | 5L6X_A |
| ✓ | Chain A, GLUTATHIONE S-TRANSFERASE [Homo sapiens] | Homo sapiens | 427 | 427 | 99% | 7e-151 | 99.52% | 209 | 4GSS_A |
| ✓ | Chain A, Glutathione S-transferase P [Homo sapiens] | Homo sapiens | 427 | 427 | 100% | 8e-151 | 99.52% | 210 | 6AP9_A |
| ✓ | Chain B, Glutathione S-transferase P [Homo sapiens] | Homo sapiens | 427 | 427 | 100% | 8e-151 | 99.52% | 210 | 5DAK_B |

(2) Use MUSCLE to perform alignments of protein sequences to identify conserved regions.


COLOR SCHEME

clustal2

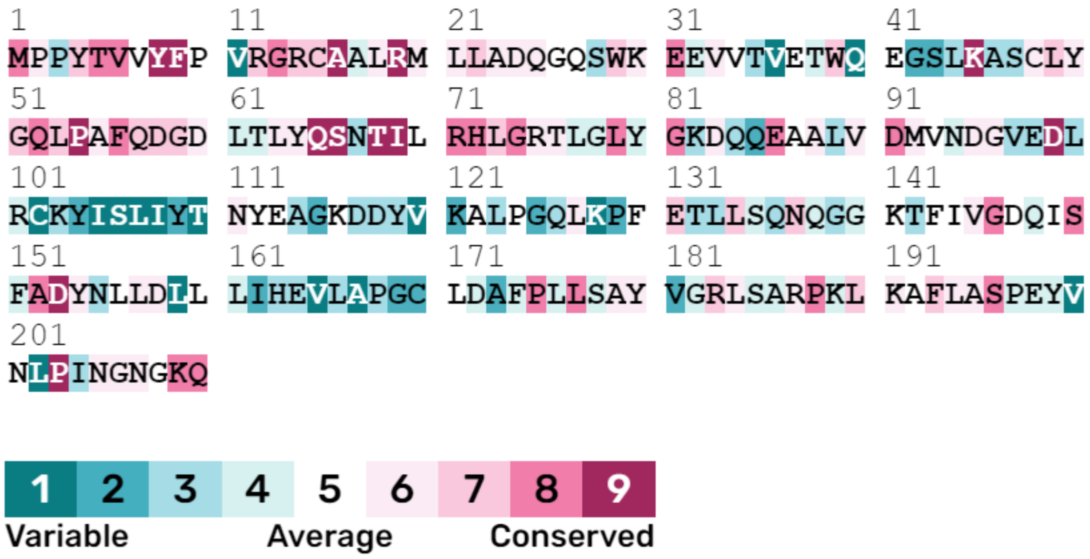
LEGEND

A R N D C Q E G H I L K M F P S T W Y V B X Z

50 sequences

XP_045414268.
XP_012520990.
XP_003941239.
XP_032138531.
XP_017392476.
1LBK_
NP_001127471.
XP_054296194.
XP_025211960.
XP_025211958.
XP_007962665.
XP_050612618.
XP_010364630.
EHH22619.
XP_011896808.
XP_011842417.
XP_011842416.
XP_030667207.
NP_001036141.
XP_023042105.
XP_011797733.
2J9H_
1KBN_
4GSS_
3KM6_
3HJM_
6AP9_
4PGT_
1EOG_

(3) Once an MSA is obtained, ConSurf and  can analyze these alignments to highlight conserved areas and predict the impact of mutations on these regions.


1 MPPYTVVYFP 11 VRGRCAALRM 21 LLADQGQSWK 31 EEVVTVETWQ 41 EGSLKASCLY
51 GQLPAFQDGD 61 LTLYQSNTIL 71 RHLGRTLGLY 81 GKDQQEAALV 91 DMVNDGVEDL
101 RCKYISLIYT 111 NYEAGKDDYV 121 KALPGQLKPF 131 ETLLSQNQGG 141 KTFIVGDQIS
151 FADYNLLDIL 161 LIHEVLAPGC 171 LDAFPLLSAY 181 VGRLSARPKL 191 KAFLASPEYV
201 NLPINGNGKQ

Molecular dynamics simulations can provide insights into how mutations affect protein stability or function by simulating the physical movements of atoms in proteins. By introducing mutations into the protein model and conducting MD simulations, we can observe changes in protein dynamics, folding, and stability. MD simulations can also assess how mutations affect the binding of ligands or interaction with other proteins, which is crucial for understanding functional impacts.

Mutational hotspots often share certain characteristics that can be leveraged to develop predictive models: (1) Regions that are critical for maintaining the 3D structure of the protein or are key to its function are often hotspots. (2) Regions on the surface that are flexible are more likely to accommodate changes, whereas rigid or buried residues might destabilize the protein when mutated. (3) Active sites or interfaces for protein-protein interactions are often susceptible to mutations that can drastically affect protein function.

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data = pd.read_csv('mutation_data.csv')  # A dataset containing mutation
impact scores, conservation scores, structural features, etc.

# Prepare the data
X = data.drop('Impact', axis=1)  # Features including conservation scores,
structural info
y = data['Impact']  # Target variable

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train a Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions and evaluate the model
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Use the model to predict new mutations
new_data = pd.read_csv('new_mutations.csv')
new_predictions = model.predict(new_data)
```