Visualization Dashboard for Rigidity Analysis of Protein Mutations

Dylan Carpenter, Josh Dombal, Hunter Read
Western Washington University
Bellingham, WA, USA

***Abstract*—Analyzing the effects of protein mutation is crucial to future treatments and development of novel medications for some diseases. However wet lab experiments can be time consuming and exhaustive screenings of protein mutations impossible. To this end, several tools, and computational software can guide this research, but may not provide overall insights into all possible protein mutations. For this, we have developed a visualization dashboard that enables easier insights into the effects of protein mutation through rigidity analysis, and is an effective tool in exploration and identification of possible critical mutations. We conclude that our visualization dashboard allows for novel insight into exhaustive protein mutation data.**

***Index Terms*—bioinformatics, mutation, protein, rigidity, visualization**

I. Introduction

Gaining insight into protein mutations is important for researchers and pharmaceutical companies looking to develop new treatments and medications. With some insight into mutant proteins, we could advance cancer diagnosis, prognosis, and therapies [2], develop novel medications for Fabry disease, or further a general understanding into many illnesses that plague civilization. Current techniques allow experimentalist to mutate and analyze these variations in a wet lab, but can require months of work with only the hope of providing the information scientists may need [3]. But due to these time requirements, thorough and exhaustive protein mutation screenings are difficult for small proteins and impossible for large proteins, arising a need for in-silico protein mutation and analysis.

While several computational approaches allow for an exploration of mutant proteins, they are frequently limited to a single mutation at a time. One technique to understand a mutant protein, would be to view a 3D structure in relation to the wild type. This however is limiting, in that human understanding from a structral visualization is limited, and finding differences between a multitude of mutant proteins is impossible. Other techniques involve exploring protein folding comparatively between the wild type and a mutant protein, but this data can be difficult to interpret when the differences are small [4]. This also does not allow for some understanding and comparative analysis in regards to an exhaustive mutation screen.

Our technique combines the novel approach of rigidity analysis, exhaustive in-silico mutation data, and a comparative metric combined into a powerful visualization dashboard to provide a unique and powerful insight into the changes between various mutations. Applying rigidity analysis to a protein allows us to gain insight into the structure of the protein. Since structure is crucial to the function of a protein, this insight can be of remarkable use. Rigidity analysis models the biomolecule as a mechanical structure, by identifying stabilizing interactions, and identifying rigid units. Then from the mechanical model, an associated graph is developed, and an efficient algorithm allows us to infer the rigid and flexible regions of this graph. This approach does not require costly energy calculations as is required by some other techniques, or require large datasets as is necessary for machine learning methods. Rigidity analysis allows for insight into protein stability, flexibility, and chemical clustering that may guide a deeper understanding for further exploration in a wet lab environment.

For this work, we have gathered computed rigidity analysis data for six proteins of various sizes and provide an interactive dashboard that allows for the exploration of mutations in regards to the wild type and comparatively for the exhaustive mutation screening.

II. Methods

A. Data

The data collected is the result of output from KINARI [1], which provides the rigidity analysis

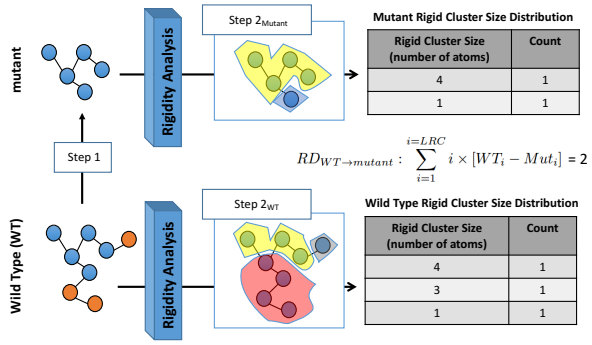

Fig. 1. Ridigity Analysis provides insight into cluster size and cluster counts for proteins. Rigidity distance can be calculated from this information with the equation given.

data consisting of cluster sizes and the counts of each cluster size as seen in Figure 1. The output of KINARI also provides information about average cluster size, degrees of freedom, and other mechanical model information. KINARI processes the data for the wild type natively, but requires additional software to handle mutations. ProMuteHT [5] provides the compute pipeline for generating mutants and SCWRL4 [6] allows for side-chain conformations to larger residues using homology modeling. With this combination of tools, the data has been generated for an exhaustive set of every possible amino acid substitution in a given protein.

However, the generated data is raw, unprocessed, and distributed among countless files. This proves rather difficult to interpret and analyze the results. The raw data files are collected together and the valuable data is extracted from the output files. Using the cluster size and count metrics, we then compute a rigidity distance metric, which compares the wild type distributions to each mutation, given by the equation $RD_{WT\rightarrow mutant}$: $\sum_{i=1}^{LRC} i \times [WT_i - Mut_i]$. This metric can provide some intuition about the protein's stability, with negative rigidity distance being a more unstable mutation, and a positive rigidity distance, more stable. All the data is gathered into a single json file for a protein and provides the foundation for the visualizations.

B. Visualization Tools

The overall dashboad was constructed using html, css, and some simple javascript. This contains the header bar that allows for protein selection and heatmap options menu. All tools were built using web frameworks which allows for an easily deployable web application, and powerful extensibility with other web frameworks for further improvements.

The heatmap was built entirely using the version 3 of the framework D3 [7]. First, we process the json data used for only for the heatmap into a 2-dimensional array of objects. This array provides the foundation of the heatmap, with the object at each index containing the values for rigidity distance, cluster information, and the mechanical model information of the mutant. The array is arranged in such a way that the columns represent the amino acid that was substituted in place of the wild type's residue, and the rows represent the numbered amino acid in the chain that makes up the protein. Of note is amino acid's that would not be mutated, as they would represent no substitution of amino acid from the wild type, which have no object values at the given index.

Then using D3, each array element is given a cell in a 2-dimensional svg grid and color scaled to the appropriate values. When a user wishes to change the heatmap options to represent a diffrent subset of data other than rigidity distance, the heatmap transitions to the new data type within the objects at each index. This allows for quick transitions without reloading the entire data set aided by the powerful transition animations from D3. Both the scale and colors update to reflect the new values during the transition, so the user experience is kept fluid and robust. D3 also provides the interactivity necessary for integrating the bar chart, updating the heatmap row and column reductions, and the tooltip information.

We used Billboard.js explicitly for the creation of the barchart. First I'll discuss how we accessed and processed our data. The original data came from a json file, but it was processed by D3 before we touched it with Billboard.js. When the data is handed off to us, it is in the form of an object that contains three arrays(mutation, sizes, counts) and each of those contains embedded arrays as well. Every time a mutation is clicked in the hashmap, that data structure is updated appropriately.

In order to use that data to create a clustered bar chart, we transform it into a matrix. The challenge here was that every time a new mutation was added, we had to create a new matrix and fill in empty values with 0 as the number of mutations grew. To create the matrix, we used nested hashmaps to map mutations to their corresponding sizes and counts. Then while iterating through the matrix, we looked up values in the hashmaps and inserted values accordingly.

Once the data was neatly formatted as a matrix, we fed that into Billboard.js and created barchart. We used .generate to create the chart, and within that call we were able to customize our barchart by using different keywords. We sized it, specified our data, the type of chart, size of the bars, x and y labels and formatting.

III. Visualizations

A. Heatmap

B. Bar Chart

We created a dynamic clustered barchart that was designed to accomplish the high level task of analysis. The goal of creating the barchart was to allow the users to consume existing information in order to draw conclusions. The existing data in our project are the counts of specific sizes of clusters in a particular mutation. These mutations can be dynamically chosen from the heatmap, and are then compared with the wild type. After selecting the specific mutations the user wants to compare, they can then come to conclusions or new hypotheses more easily by being able to compare differences side by side.

The marks that are used are vertical bars; which represent the different sizes of clusters in the chart. We used multiple channels when designing our barchart. We used position on a common sale because it is very effective to compare different values. We also used color hue so that the user can easily identify the different mutations on the chart simply by looking at the colors. We also made use of horizontal position in order to separate the different sizes of clusters. We used multiple channels so that the user can easily look at the chart from different angles and analyze the chart for different purposes.

Figure ???? shows the basic chart before any mutations have been added from the heatmap.

IV. Findings

V. Discussion

VI. Conclusions

References

[1] N. Fox, F. Jagodzinski, and I. Streinu. "Kinari-lib: a C++ library for pebble game rigidity analysis of mechanical models." In Minisymposium on Publicly Available Geometric/Topological Software, 2012.

[2] Q. Wang, R. Chaerkady, J. Wu, H. Hwang, N. Papadopoulos, L. Kopelovich, et al. "Mutant proteins as cancer-specific biomarkers". Proceedings of the National Academy of Sciences. 2011.

[3] T. Maximova, R. Moatt, B. Ma, R. Nussinov, and A. Shehu. "Principles and overview of sampling methods for modeling macromolecular structure and dynamics." PLoS computational biology. 2016.

[4] A. Oliveira Jr., F. Fatore, F. Paulovich, O. Oliveira Jr., and V. Leite "Visualization of Protein Folding Funnels in Lattice Models." PLoS computational biology. 2014.

[5] E. Andersson and F. Jagodzinski. "ProMuteHT: A high throughput compute pipeline for generating protein mutants in silico." In Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics 2017.

[6] G. Krivov, M. Shapovalov, and R. Dunbrack. "Improved prediction of protein side-chain conformations with scwrl4". Proteins: Structure, Function, and Bioinformatics, 2009.

[7] M. Bostock, V. Ogievetsky, and J. Heer. "D3 Data-Driven Documents". IEEE Transactions on Visualization and Computer Graphics, 2011.