An Exploration of Dimensionality Reduction Techniques as Applied to Mutated HIV-Vif-A3F Molecular Dynamics Simulations

Nina van Hoorn

Skidmore College
Computer Science and Chemistry Departments
April 18, 2025

Table of Contents

ABSTRACT 3
1. INTRODUCTION 4
2. RUNNING THE MOLECULAR DYNAMICS SIMULATIONS 8
3. MAKING THE K50E VIF MUTATION 10
4. IDENTIFYING CONFORMATIONAL CLUSTERS USING PTRAJ PCA 14
5. FURTHER EXPLORATION OF PTRAJ PCA 25
6. VISUALIZING COMPLEX MOTIONS USING SCIKIT-LEARN PCA 34
7. REDUCING DIMENSIONALITY THROUGH KERNEL PCA 42
8. VISUALIZING THE LATENT SPACE THROUGH AN AUTOENCODER 48
9. THE EFFECTS OF THE MUTATION 56
10. CONCLUSIONS 60
ACKNOWLEDGMENTS 62
GLOSSARY 63
REFERENCES 64

Abstract

Viral infectivity factor (Vif) is an intrinsically disordered protein used by HIV-1 to hijack an immune cell's natural defenses. Vif binds to an E3 ubiquitin ligase complex composed of Elongin B, Elongin C, CBF-ß and Cullin 5, allowing Vif to act as a substrate binding protein for antiviral APOBEC3 proteins (A3s), resulting in their ubiquitination and degradation. This systematic removal of the A3s allows HIV-1 to more easily spread to other cells without encountering an antiviral response. Understanding how Vif interacts with the host proteins could lead the way to developing counteractive treatments to prevent binding to the A3s and the subsequent degradation. The cryogenic-EM structure of A3F (one member of the A3 family) bound to Vif and CBF-ß has been used in molecular dynamics (MD) simulations to model these unknown interactions between Vif and A3F. These MD simulations were able to identify the specific residue interactions in the Vif-A3F binding domain that stabilize the conformations sampled by the complex, and free energy analysis was used to identify the residue that became most stabilized after binding. This lysine residue was mutated to glutamate, and then more MD simulations were run to simulate how the mutated version of Vif would interact with A3F. Using both the wild type simulation data and the mutated simulation data, a variety of dimensionality reduction techniques were explored to compare how these two simulation types map to lower dimensional spaces. Techniques included principal component analysis (PCA), mapping data to a feature space using kernel functions and rerunning PCA, and using a trained autoencoder to compress the data. These mappings revealed that the mutation of the lysine residue caused changes to the protein's dynamics, hinting at the importance of the identified residue for stabilizing Vif-A3F binding. Additionally, this research could lead to new insights on Vif's affinity towards A3F, potentially aiding the development of treatments that can target and interrupt Vif-A3F binding, rendering Vif and the rest of the virus useless.

1. Introduction

Intrinsically disordered proteins (IDPs) are highly dynamic, flexible proteins that lack a stable folded structure. They have also been found to be an important component of various cancers, diabetes, diseases and viruses, and are thus considered potential drug targets to help fight these conditions[1,2]. Human Immunodeficiency Virus Viral infectivity factor (HIV-Vif) is an intrinsically disordered protein utilized by the virus HIV to help prevent an antiviral response in the human body. HIV is a virus that has infected over 70 million people and causes the progressive deterioration of the immune system which ultimately leads to death if left untreated[3,4]. Currently, there are treatments for people living with HIV that can help prevent death and improve quality of life, but there is no total practical cure[5].

Vif is a crucial part of HIV that makes it easier for the virus to infect an organism. In order to prevent an antiviral response, Vif hijacks the E3 ubiquitin ligase complex composed of Elongin B (ELOB), Elongin C (ELOC), CBF-β, and Cullin 5 (CUL5). In a healthy cell, this complex will have a suppressor of cytokine signaling protein (SOCS)2 which is an adaptor protein that determines which proteins will become substrates for ubiquitination. Once bound, this complex covalently attaches ubiquitin to the substrate, marking that protein for degradation. However, when Vif hijacks this complex, it replaces SOCS2 as the adaptor protein, effectively gaining the ability to determine which proteins get marked with ubiquitin, and subsequently degraded (Figure 1). In order to increase its stability, Vif also recruits Core binding factor subunit, CBF-β, which is vital as it inhibits Vif oligomerization and activates CRL5-Vif[6].

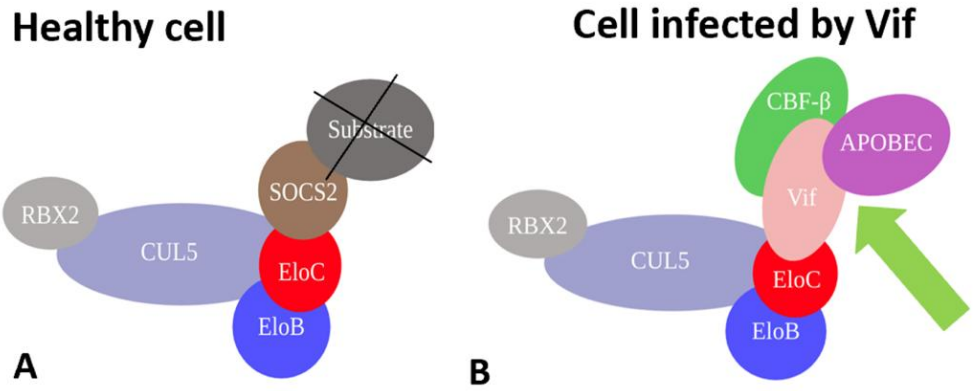


Figure 1. (A) The E3 ligase ubiquitination complex found in a healthy cell and (B) the complex when hijacked by Vif. In a healthy cell, SOCS2 would determine which substrates get marked with ubiquitin, but after Vif hijacks this complex, it gains the ability to mark APOBEC proteins with ubiquitin. CBF-β is recruited to stabilize the hijacked complex. From ref 7.

Once bound to the E3 ubiquitin ligase complex, Vif binds APOBEC3 (A3) proteins, which are essential proteins in the immune system (the whole complex with A3F bound can be seen in Figure 2)[8]. The A3s are a family of DNA cytosine deaminases that provide protection against a wide variety of pathogens, and four of them (APOBEC3D (A3D), APOBEC3F (A3F), APOBEC3G (A3G), and APOBEC3H (A3H)) actively work to prevent HIV-1 replication[9]. However, as a result of this targeting by Vif, the A3s are marked for ubiquitination and degradation. The systematic destruction of these antiviral proteins facilitated by Vif allows HIV to infect the body without facing a strong antiviral response[10]. An increased understanding of how Vif and A3F bind could be useful for the development of pharmaceuticals which target the IDP at this binding site; if binding could be inhibited, A3 proteins would not be degraded and the A3s could prevent the replication of HIV.

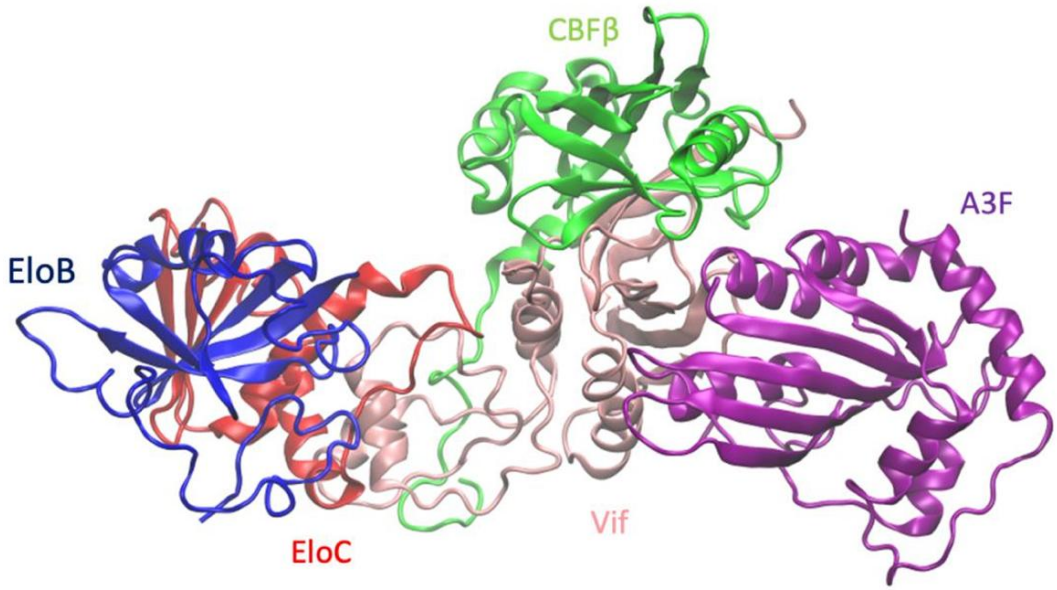


Figure 2. The VCBC-A3F complex visualized in VMD. Vif is shown in pink, A3F in purple, CBF-ß in green, EloC in red, EloB in blue.

Due to the intrinsically disordered nature of Vif, traditional experimental methods of studying protein structure, such as nuclear magnetic resonance (NMR), are insufficient as they are unable to reflect the ensemble of conformations the IDP will adopt due to its flexible nature[1]. As a result, computational methods have often been used to better understand the behavior and interactions of IDPs. Molecular dynamics (MD) simulations are one such method. They capture the behavior of proteins in full atomic detail over a period of time by predicting the movement of every atom in small timesteps using a general model of the physics affecting the interatomic interactions[11]. We have previously used MD simulations to study the VCBC complex and learned that it undergoes conformational changes and that a flexible linker region in Vif is fundamental to these changes[7]. We also determined that the complex samples more states when Cul5 was not bound. These are all findings that would not be possible to learn from just studying

stationary proteins using traditional experimental methods. For the scope of this paper, we've run two different sets of MD simulations (one wild type simulation and one with a single residue mutated) with A3F bound to the VCBC complex in order to study the Vif-A3F interaction. The information gathered from these simulations can provide key insights into how the proteins move, function, and bind. While the simulation data contains a plethora of information about the proteins and their interactions, the high-dimensional nature of the data makes it difficult to know how to best analyze the data that emerges from MD simulations. To extract useful information from the large amount of high-dimensional data, dimensionality reduction is required[12,13].

One feature that dimensionality reduction techniques can help highlight is conformational changes. The functionality of a protein is not just dependent on its structure; proteins are dynamic, especially IDPs, and move between conformational states, with different ensembles that change with the environment[14]. A3F might have a different affinity for binding to different conformations of the VCBC complex, and therefore understanding how the complex moves between different conformational ensembles might reveal key insights into the binding process[7]. Therefore, in this paper I have used a variety of dimensionality reduction techniques on VCBC-A3F simulation data to better understand the conformational changes the complex goes through. More specifically, I compare the conformational changes of the simulated wild type (WT) to a computationally mutated version of the complex. By comparing the differences in the ensembles between the two, I hope to gain a better understanding of how the mutation impacts the binding affinity between Vif and A3F. If binding is inhibited, the residue I chose to mutate on Vif could be a target for therapeutics that could prevent the ubiquitination of A3F and the subsequent spreading of HIV.

In chapter two, I outline how the starting VCBC-A3F structure is built by combining a crystal VCBC structure and a cryo-EM structure of Vif, CBF-ß, and A3F. I then describe the constraints and specifications of the molecular dynamics simulations that were run using the created starting structure. In the following chapter, I outline how free energy analysis was used to identify Vif residue K50 as a potential target for a mutation to disrupt the Vif-A3F binding site. This residue was then mutated from lysine to glutamate to change the charge, and eight independent simulations were run with this mutated starting structure.

In chapter four, I begin using principal component analysis (PCA) to analyze the wild type and mutated simulations. PTRAJ software was used to carry out this process; PRAJ is a powerful tool for PCA as it enables the visualization of the principal components to provide structural meaning in terms of the VCBC-A3F complex. After projecting the simulation data onto the top four principal components, I use k-means clustering to help identify differences between the two groups of simulations. I then continue exploring PCA in chapter five as I look at the principal components found using just the wild type simulations or just the mutated simulations, as opposed to the two combined. By doing this, I was able to realize that something

in our methodology was wrong due to unexpected standard deviations for the top ten principal components. This implies that the PCA from chapters four and five are likely not accurate or as informative as initially believed, but these chapters still provide a framework of how this analysis can be carried out and interpreted in the context of analyzing the global motions and structural conformations sampled by the complex throughout simulation time.

As I explored the potential error from the PTRAJ PCA, I reran PCA using the Scikit-learn library in Python, which is outlined in chapter six. This was done to provide a standard of comparison for how the PCA should look and what the ordering of the standard deviations for the principal components should be. While the Scikit-learn library is not as tailored to this line of research as PTRAJ is (with regards to PCA, there is no easy way to interpret the meaning of the principal components found by Scikit-learn), it does not appear to have the potential errors described in chapter five. Additionally, this PCA appears to highlight some different clusters that were formed from the two simulation types, though future study would be required to confirm the presence of these clusters and attempt to characterize them. Though PCA is a powerful tool to reduce the dimensionality of molecular dynamics simulation data and visualize some of the key features, it fails to identify non-linear patterns. So, in chapter seven, I explore kernel PCA (KPCA) as a way to visualize the high dimensional space. KPCA uses a kernel function to map data to a feature space where linear patterns may become apparent in the data and then runs regular PCA on this feature space. This method is capable of identifying non-linear patterns in the data, but it can also be harder to interpret and is less commonly used to analyze molecular dynamics simulations. Due to this fact, I was not able to ascribe conformational meaning to the differences I observed after running KPCA, but the presence of differences between the wild type simulation data and the mutated simulation data could be an indicative of a future direction of study.

In chapter eight I discuss the process of training various autoencoders to compress simulation data to a three-dimensional latent space. By visualizing this latent space, we are again able to identify patterns and differences between simulations that may not be apparent in higher dimensions. Like KPCA, though, using an autoencoder to compress data can be hard to interpret, so I am not currently able to interpret what the differences between the simulations projected onto this latent space mean. However, the presence of any differences at all does imply that the mutation had some impact on the structure throughout simulation time. Chapter nine summarizes the results of mutating Vif residue K50 and the potential impacts this had on simulations. This chapter includes the main results from the previous chapters as well as additional contact maps that were made to measure the contacts between Vif and A3F at the binding site. From these, we can see that the mutation decreases contacts between Vif and A3F, implying that it inhibits binding. If this inhibition is disruptive enough, it might mean that a therapeutic which targets the mutated residue could effectively prevent the replication of HIV in the body.

2. Running the Molecular Dynamics Simulations

To study the motions and interactions of the VCBC-A3F complex, we used molecular dynamics simulations. These simulations capture the location of every atom in the complex over the course of 300 ns. In total, we ran 16 simulations; eight simulations were run using the wild type starting structure and eight were run using a computationally mutated starting structure. By running these two sets of simulations, we were then able to compare the two to see the impacts of the mutation of the behavior of the complex as a whole. By doing this, we were able to better understand the importance of the site of the mutation for the binding of Vif and A3F.

In order to obtain simulation data for the wild type of VCBC-A3F, we created a starting structure for the simulations combining the cryo-EM structure containing Vif, A3F and CBF-B (PDB number 6NIL) with the crystal structure of the VCBC-Cul5 complex (PDB number 4N9F)[15,16]. To combine these two structures and make VCBC-A3F, Cul5 was removed from the crystal structure and the Vif residues were aligned in VMD, since Vif was present in both structures[17]. Once they were aligned, the cryo-EM structure was added to the VCBC structure. In the combined structure, residues 1-114 and 158-176 of Vif (which interact with CBF-B and A3F) were derived from the cryo-EM structure, and residues 115-157 (which interact with EloC) were from the crystal structure. After the combined structure was created, residues 1-16 in A3F were removed to match previous nuclear magnetic resonance experiments. A3F residues which were missing in the cryo-EM structure were built using Modeller version 9.23[18]. This software determines a protein's optimal structure given its sequence in comparison to its alignment with a similar structure. Multiple models were created using this software, and the one with the lowest Discrete Optimized Protein Energy (DOPE) was chosen to include in the starting structure as the low DOPE score indicated that model was most closely aligned to the previous known structure. This was also aligned in VMD, and once combined, the resulting PDB included the whole, complete structure of the VCBC-A3F complex

MD simulations using this VCBC-A3F starting structure were then run using the AMBER99SB force field, TIP3P water, and the AMBER24 software package[19,20]. The AMBER99SB force field was chosen to match previous simulations from our lab, and the TIP3P water box is often used in conjunction with this force field. Zinc ions were treated with a four-point charge representation and all simulations were performed on GPUs[21]. The particle mesh Ewald procedure was used to handle long-range electrostatic interactions with a nonbonded cutoff of 9 Å[7]. Tleap (a helper program that generates a topology file and restart file given coordinate files such as PDB) was used to create a parameter file and an input coordinate file from the PDB[19,22]. The parameter file contains the order of the amino acids in the proteins, the elements' size and mass, and the atomic structure and charge of each residue. The input coordinate file contains the x, y, z coordinates of the system.

Given these starting files, two sets of minimizations were run to allow the proteins attain a more stable structure (since the starting structures obtained experimentally were derived from proteins in different conditions, this helps the simulations reflect a more realistic protein). The first minimization was run with the proteins restrained at 500 kcal/mol with a 500-step steepest descent followed by a 500-step gradient minimization (this allowed the surrounding solvent to minimize). The second had no restraints. The system then was heated from 0 to 300 K over 20 ps with protein restraints at 10 kcal/mol to bring the system to roughly room temperature. Afterwards, the system was equilibrated twice; in the first equilibration the protein was restrained (with 1 kcal/mol strength) to get the solvent to equilibrate to the pressure and temperature. There were no restraints in the second, allowing the protein conformation to equilibrate. The Andersen thermostat was used to keep the simulations' temperature constant at 300 K and the Berendsen barostat with isotropic position scaling and a relaxation time of 1 ps held the system at 1 bar. The particle mesh Ewald procedure was used with a nonbonded cutoff of 9 Å.

After these initial steps were completed, eight independent simulations were run using the same equilibrated starting structure, but each with randomized initial velocities. Running eight separate simulations helps provide us with a large sample size of simulation data to analyze. Normally, these simulations would each be run for 600 ns and only the last 300 ns would be used for analysis to ensure that the system was sufficiently equilibrated. However, due to time constraints, each simulation was run for only 300 ns, and all the data was used in analysis. Consequently, some of the data included in the analysis might not be reflective of a stable structure in a low energy conformation. However the alternative was to use less data, which would mean fewer conformations would be sampled, so I decided to prioritize simulation duration and data quantity over increased equilibration time. Because some of the simulation data might not be reflective of an equilibrated structure, the conformations sampled in the simulation data might not be sampled by the complex in natural conditions. Consequently, some of the results seen might be a result of the high energy conformations undergoing further equilibration. Additionally, not having enough equilibration time could result in the wild type simulations and mutated simulations appearing more similar than they would be after equilibrating fully.

3. Making the K50E Vif Mutation

Free energy analysis was used to determine which residues of Vif were the most stabilized by the presence of A3F in the simulations. The residues that are more stable with A3F bound are also the residues that likely facilitate binding in order to increase stability. By pinpointing one specific residue located at the binding site that was stabilized by A3F's presence, we were able to choose a residue to mutate for our next round of simulations. If, as a result of this mutation, this residue was not stabilized by A3F binding, the binding site could be negatively impacted, and A3F could even be prevented from binding to Vif. This would mean Vif would be incapable of causing A3F's degradation and HIV could be fought off with an immune response. Therefore, the spot of this mutation would be a potential target for future therapeutics that would render HIV defenseless against the body's immune system.

A similar procedure as that described in the previous chapter was followed for the VCBC complex without A3F bound[7]. Using simulation data for VCBC-A3F and the data for VCBC, we were able to calculate the differences in free energy between the two complexes using Molecular mechanics [MM] with Poisson–Boltzmann [PB] and surface area solvation (MMPBSA)[19]. This is an analysis technique that calculates the free energy contribution of each residue at each time point; it takes into account the average internal energy, Van de Waals forces, electrostatic interactions, polar solvation, and non-polar solvation. Entropy contribution is not taken into account except implicitly in the solvation terms. Higher free energy equates to less stability, and low free energy means a highly stable residue. After calculating the free energy in the simulation with and without A3F, we subtracted the free energies to get the free energy difference at each residue. This was used to identify the residues of Vif that had the largest decrease in free energy when A3F was bound compared to the simulations without A3F. The residues with the biggest difference in free energy are those that are most impacted by the binding of A3F to the complex and if the free energy decreases with A3F binding, these residues likely play a role in stabilizing the binding interaction. Vif residue K50 was shown to have the second biggest decrease in free energy (13.22 kcal/mol). Unlike the residue with the highest decrease, K50 is located at the binding site between Vif and A3F; this residue is in contact with A3F for 100% of simulation time. It is a positively charged lysine and forms a salt bridge with A3F residue E103, a negatively charged glutamic acid (Figure 3). The large decrease in free energy, the location, and the salt bridge formed by this residue all indicates that Vif residue K50 is an important residue that stabilizes the binding of Vif and A3F, and therefore this was the residue we chose to mutate.

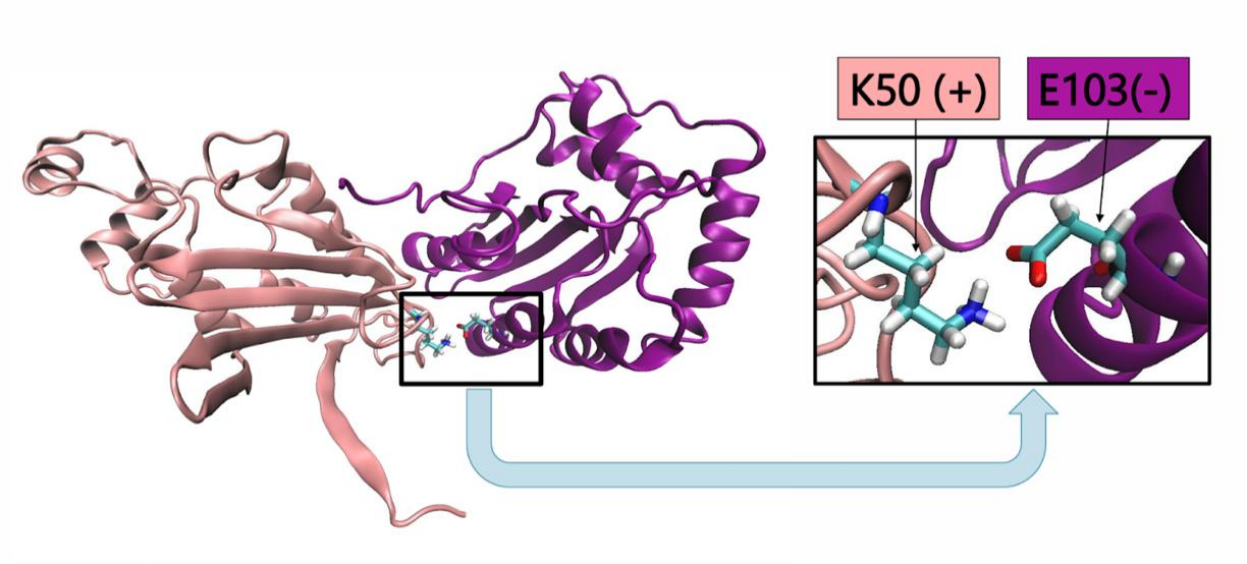


Figure 3. VMD image with Vif residues (pink) and A3F residues (purple) shown. Vif residue K50 forms a salt bridge with A3F residue E103 at the binding site, so the residue was mutated to glutamate to disrupt binding between Vif and A3.

In order to disrupt the interaction between the positive lysine in Vif and the negative glutamic acid in A3F, I decided to mutate the lysine to a glutamic acid, as this would switch the charge of the residue without changing its structure significantly (Figure 4). This would create an unfavorable interaction between these two residues (since the two negative charges would repel each other) and potentially hinder the binding process. In order to make sure this mutation would only affect the interaction with A3F residue E103, I first checked the distance between Vif residue K50 and all other surrounding charged residues using VMD. In Vif, there were only three charged amino acids nearby, the nearest one being 11.74Å away. I decided this distance was big enough that changing the charge of Vif K50 would not cause problems within Vif.

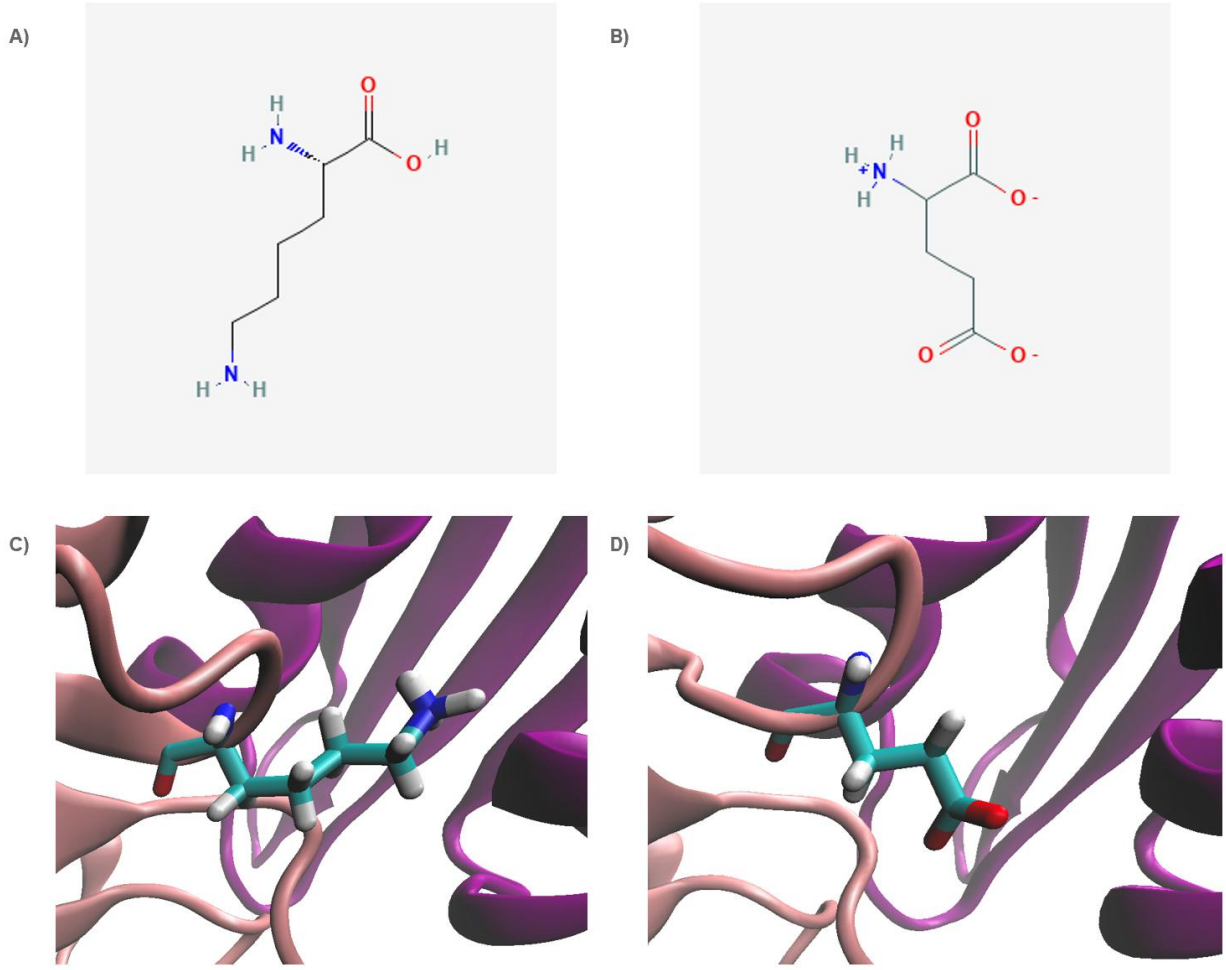


Figure 4. A) Chemical structure of lysine and B) glutamate[23,24]. C) VMD image of Vif residue 50 (lysine) from the wildtype simulations and D) VMD image of Vif residue 50 (glutamate) from the mutated simulations

To make the mutation, I first wanted a good starting structure. Rather than making the mutation directly to the same VCBC-A3F PDB that was used for the previously described simulations, I decided to use a starting structure in a more stabilized form derived from the simulation data. I first used AMBER to find the average structure of the protein from the last 300ns of all 8 simulations. This average structure best represented the most common conformation of the protein throughout the simulations, but I did not want to use an averaged structure as a starting point for new simulations as it could be unphysical. I therefore compared the RMSD of each frame of the simulations to the average structure and used the frame with the lowest RMSD as my new starting structure. After converting this to a PDB, I changed Vif residue K50 from “LYS” to “GLU” and removed all elements that were not a part of the amino acid backbone. Tleap was then used to generate the new PDB with the mutated residue, the new parameter file and the new input coordinate file. Then, I followed the same procedure to stabilize

the protein and solvent (two minimizations, heating, and two equilibrations) and ran the simulations. After this, I had eight independent simulations of the mutated VCBC-A3F complex to compare to the wild type simulations.

4. Identifying Conformational Clusters using PTRAJ PCA

As previously explained, MD simulations can be hard to interpret due to their high-dimensional nature. For the wild type simulations of the VCBC-A3F complex, there are 480,000 frames of 36,180-dimensional data, and for the mutated simulation data there are the same number of frames but 36,159-dimensional data. The difference in dimensionality comes from the slight difference in number of atoms in the simulations; glutamate has fewer atoms than lysine, and the number of dimensions comes from three multiplied by the number of atoms, with the three representing x, y, and z coordinates in space. So, in total there are 960,000 data points falling somewhere in the 36,000-dimensional space containing information about the protein's configuration in space at that point in simulation time. This is a large amount of data and many of those 36,000 dimensions are likely redundant, which is the motivation for the dimensionality reduction. For the techniques I explored though, I only used the atoms that were a part of the protein backbone. This restriction allowed the two types of simulations to have the same number of atoms and took less computational power and memory to analyze. The conformation of the complex is also determined by the backbone, and thus the additional atoms were not necessary to include in the structural analysis. As a result of this decision, the dimensionality of the data being analyzed was 8,904 dimensions with only 2,968 atoms. Additionally, clustering the data can help reveal important patterns in the data; if the simulation data falls into multiple, separate clusters, characterizing these clusters can help us better understand the conformations that the protein moves through during the simulations. However, in order to cluster the data, reducing the dimensionality of the data is required to eliminate redundancy and highlight the most important dimensions of the data for analyzing.

One common method of linear dimensionality reduction is Principal Component Analysis (PCA). PCA reduces the dimensionality of a high dimensional dataset by projecting the data onto a lower dimensional linear subspace that maximizes the variance of the projected data[25]. Due to the high dimensional nature of MD simulation data, PCA has become a widely used method to dramatically decrease the dimensionality of the configurational space, which has aided with further analysis. It is also widely used because it is relatively simple and less computationally expensive than other potential methods[26]. Previous studies have found that, despite the relative simplicity of PCA compared to other dimensionality reduction techniques, the method still holds up and provides an informative visual depiction of the high dimensional space. However, in certain situations, PCA cannot perform as well as some other techniques, which is mainly due to its linear nature; if the data does not have strong linear patterns, PCA will not be able to identify any patterns because it projects the data onto a linear manifold (Figure 6)[27]. By applying PCA to my data, I am able to not only identify the *n* linear dimensions that have the highest data variance (consequently removing dimensions that are less useful or redundant), but I am able to visualize

how the mutant simulation data falls along these dimensions and compare that to the wild type simulation data.

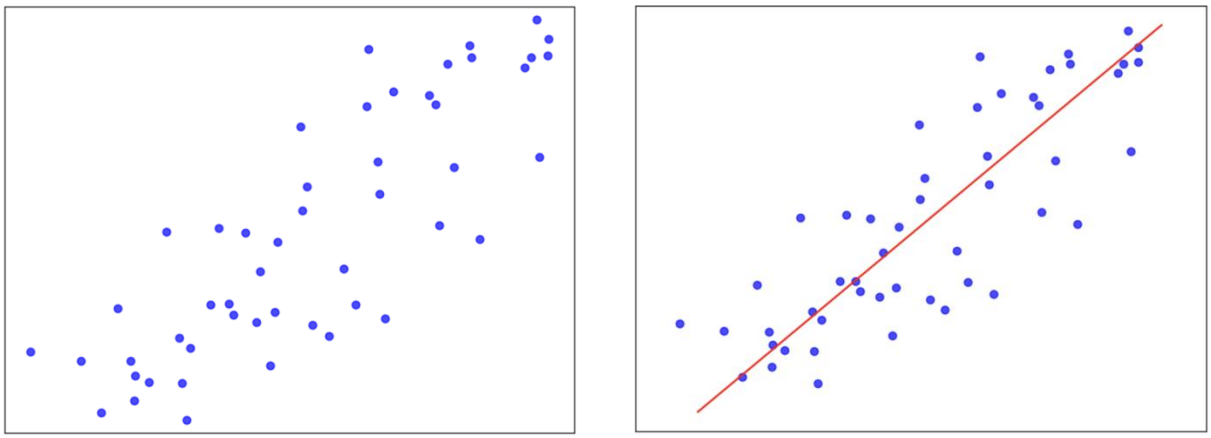


Figure 5. Given a spread of data (shown on the left), the first principal component is the vector (visualized by the red line on the right) that, when all the data points are projected orthogonally onto it, has the highest variance.

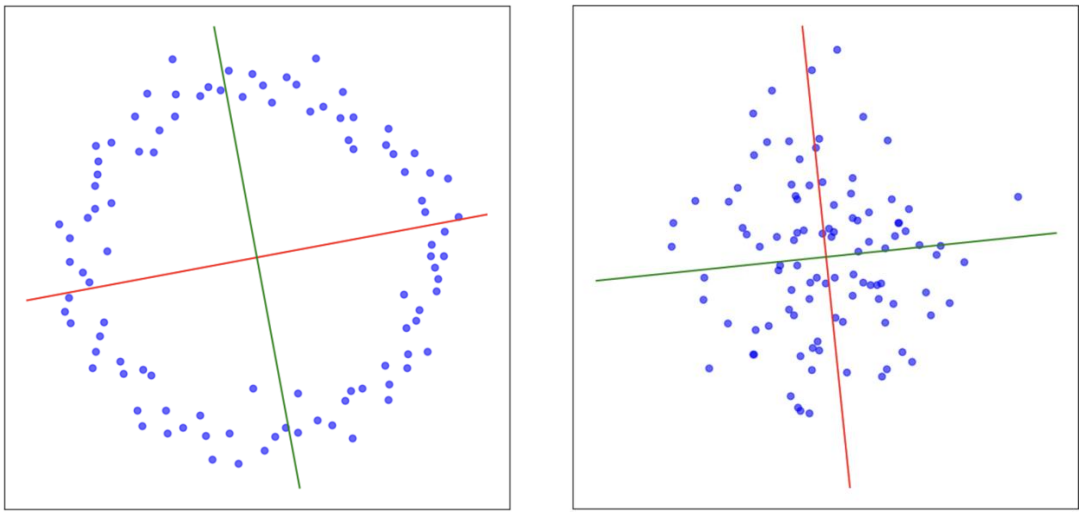


Figure 6. Given the two spreads of data shown above, simple PCA will not be able to find meaningful linear patterns in either (the first principal components for both spreads of data are shown by the red line and the second is the green line), yet there is a clear pattern in the data on the left.

In order to identify the 10 linear dimensions that contain the highest data variance, I used PTRAJ, a computer program used to process and analyze three-dimensional atomic position time series[28*]. First, the data from the wild type simulations and the mutated simulations were loaded

* Relevant code for analysis is available on GitHub at *https://github.com/Skidmore-Computational-Biophysics-Lab/Nina_van_Hoorn_Thesis_Files*

in; both were included to generate the principal components (PCs) so that the PCs would reflect the conformational ensembles of both the complexes. This allowed the two sets of data to be projected onto the same PCs and directly compared. (Every tenth frame of all of the simulations were used in order to save time and computational power; since subsequent frames are nearly identical, using every tenth frame should not strongly impact results.) I then used the first frame from the first mutated simulation as a reference to align the backbone residues for all the simulation frames by minimizing the coordinate root-mean-squared deviation (RMSD) compared to the first frame[29]. Using the aligned structures, I then generated an average structure based on the average of the coordinates for the backbone atoms over all frames. This average structure was used to once again align all the structures to the average structure to ensure all the simulations were oriented in the same way in space. I then generated a coordinate covariance matrix, which contained the covariance between each of the frames, based on the coordinate positions.

Once I had this covariance matrix, I was able to calculate the top ten eigenvectors through diagonalization. To ensure that most of the data variation was captured in the top ten eigenvectors, I also calculated the top 100 eigenvectors through the same process and was able to check the fractions of the total data variation captured in the top 10 vectors. This relies on the assumption that approximately 100% of the data would fall in the top 100; Table 1 shows the top seven of these 100.

Table 1. Percentages of the total data variation captured in the top seven eigenvectors

Principle Component	Percentage of total variance	Cumulative percentage
1	21.15%	21.15%
2	14.95%	36.10%
3	12.11%	48.20%
4	6.79%	55.00%
5	5.51%	60.51%
6	3.97%	64.48%
7	3.60%	68.08%

Fifty-five percent of the total variance in the data is captured in the top four principal components; though there is a large amount of data variance not reflected in these PCs, it is still majority. Therefore, I determined that analysis performed using the top four PCs could help inform our understanding of the global motions of the proteins. After this, I saved the top 10 eigenvectors and projected my simulation data onto these dimensions. The top four PCs are shown in Figure 7; the first appears to be folding of the complex, the second a clamshell type of motion of the VCBC part of the complex, and the third and fourth seem to be two types of twisting motions. The projections of the simulation data onto these PCs are shown in Figure 8 with the wild type simulations and the mutated simulations plotted separately.

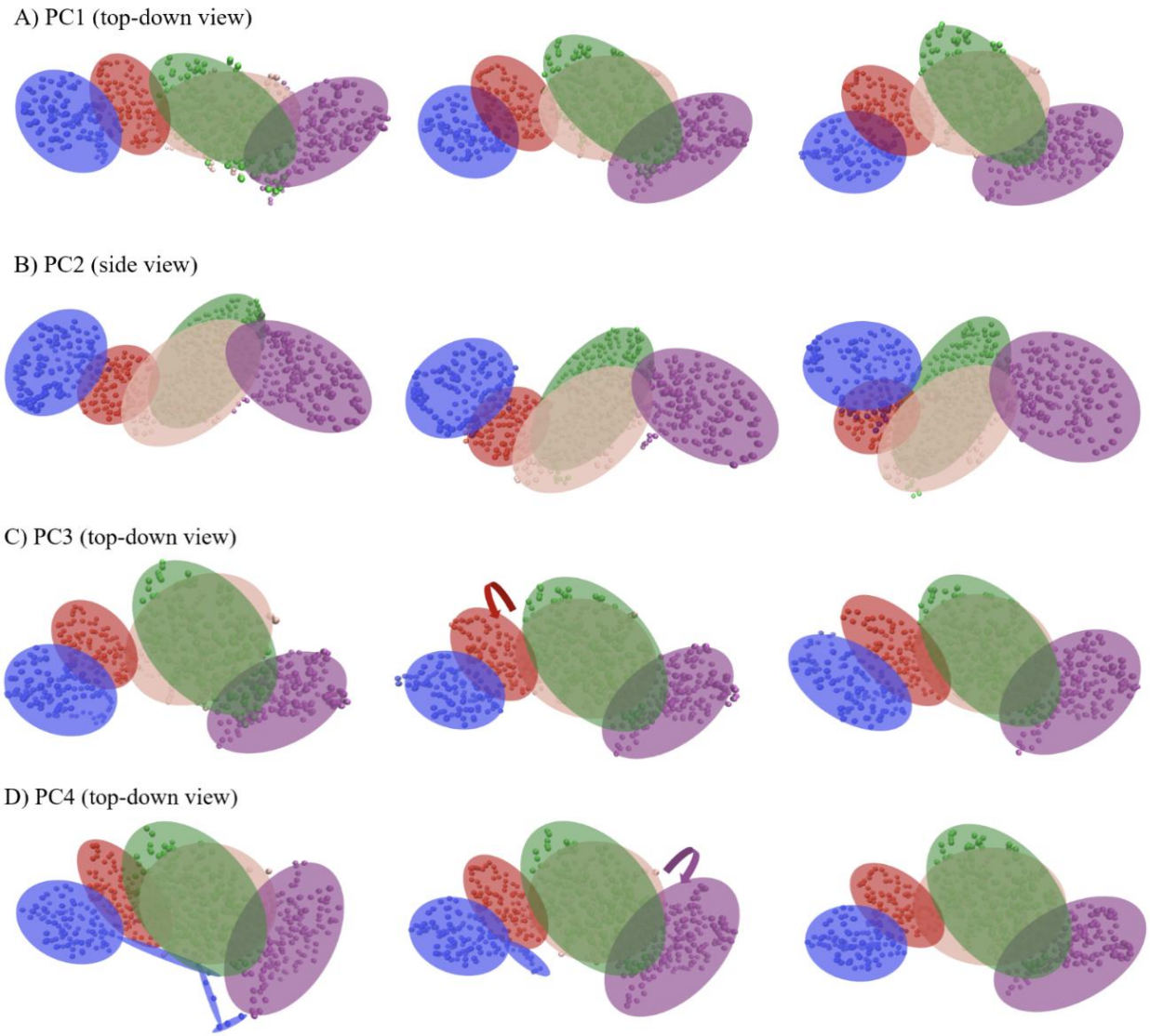


Figure 7. Visualizations of the top four PCs from left to right; videos showing the movements can be seen here; the proteins from left to right are (blue) EloB, (red) EloC, (green) CBF-ß, (pink) Vif, and (purple) A3F. A) PC1 starts with all of the proteins lined up and, over time, folds to one side, bringing A3F and EloB closer together. B) PC2 appears to be a clamshell motion, with EloB and EloC coming up to meet Vif and CBF-ß. C) In PC3, EloB and EloC appear to be twisting together while the rest of the complex stays relatively stable other than a slight closing motion, similar to the clamshell one. D) In PC4, most of the complex stays together, but A3F twists in the opposite direction. There is also a long tail on EloB that initially appears to interact with A3F, but recedes as PC4 changes.

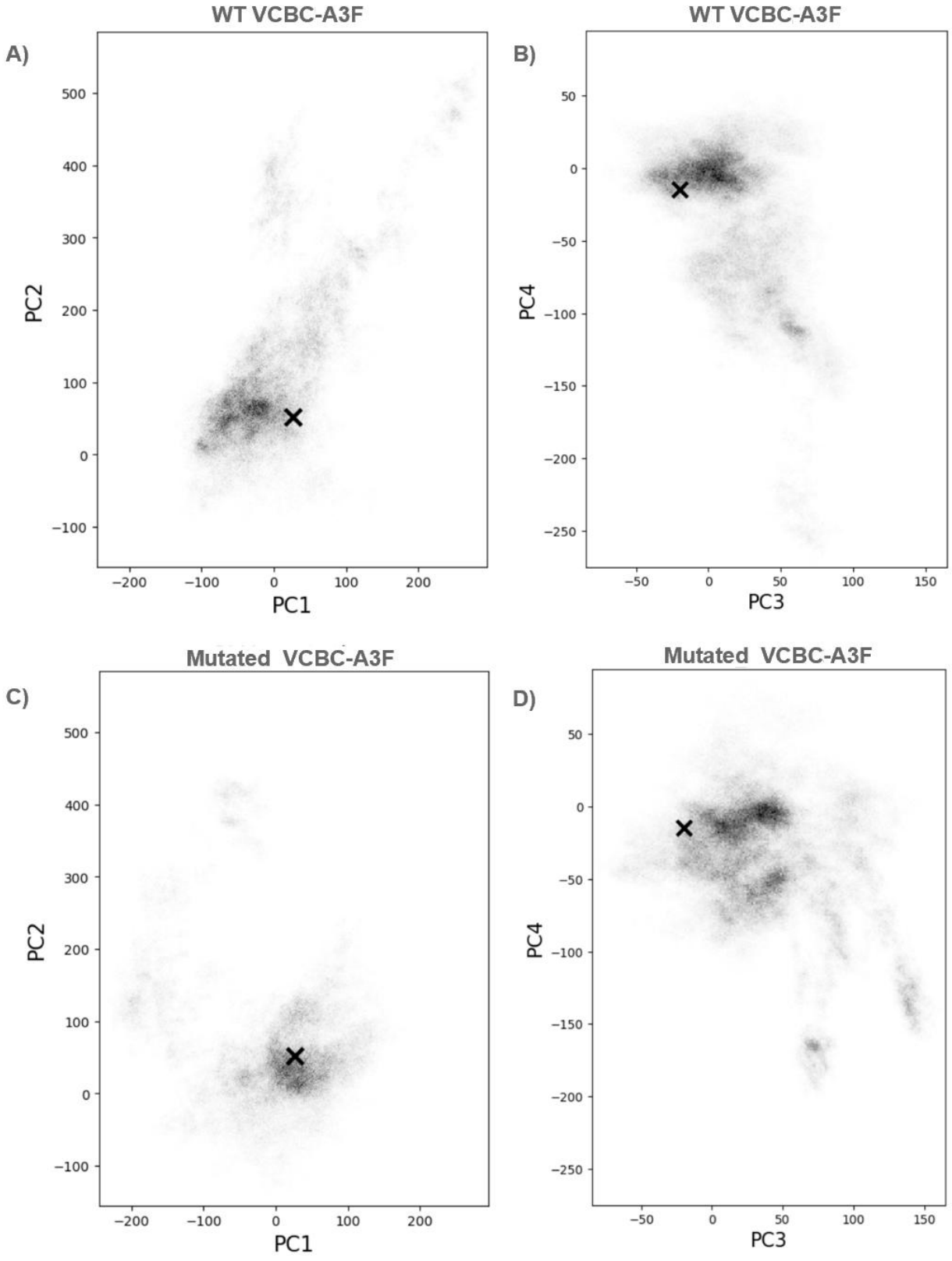

A)
WT VCBC-A3F
PC2
PC1
B)
WT VCBC-A3F
PC4
PC3
C)
Mutated VCBC-A3F
PC2
PC1
D)
Mutated VCBC-A3F
PC4
PC3

Figure 8. A) The wild type simulation data projected onto the first two PCs and the B) third and fourth PCs. C) The mutated simulation data projected onto the first two PCs and the D) third and fourth PCs. The black X represents the starting structure for the simulations; the darker areas in the plots correlate to more simulation time spent in that conformation and lighter areas were sampled less in the simulations.

Table 2. Standard deviations and means of simulation data for along top four PCs.

		PC1	PC2	PC3	PC4
Standard Deviation	Wild Type	67.86	115.38	30.92	53.60
	Mutated	76.63	88.96	42.90	44.61
Mean	Wild Type	0.23	114.51	16.70	-36.93
	Mutated	-0.55	61.40	36.72	-38.30
P-value Difference in Variance		0.163	0.302	0.587	0.079

** Some of these standard deviations, such as the PC2 variances being relatively high were unexpected. This is indicative to us that there may be something wrong with how these are calculated, which I'll address more in the next chapter.*

It appears that along PC1, the mutated simulation data stays closer to the conformation of the starting structure than the wild type data, which spends most of its time in a more linear (less folded) conformation. Despite this, the wild type simulation data also has more data which falls into a folded conformation, shown in the diagonal spread of data moving up and right from the main concentration of data in Figure 8(A). However, the mutated simulation data has more variance in PC1 in general with a standard deviation of 76.63 compared to a standard deviation of 67.86 in the wild type data. Though this difference was observable, it was not statistically significant with a p-value of 0.163. From the figure, the variation in the mutant conformation seems to sample more toward the linear conformation; the mutated simulation data never appears to have high PC1 values, meaning it never becomes as folded as the wild type simulation (at least in the folding motion described by PC1).

Along PC2, the wild type data has much more variance than the mutated data with a standard deviation of 115.38 and a large streak of data going upwards into the "closed clamshell" conformation. Though the simulation data is still concentrated near the starting structure at relatively open conformation, this high variance could indicate that the mutation somehow inhibited the complex's ability to close. The mutated data had a standard deviation of 88.96 for this PC and appears to be almost entirely concentrated around the starting structure other than one area falling at about the 400 mark on the PC2 axis in Figure 8 (C). The p-value calculated for the difference between the wild type simulation and mutated simulation data variance along this principal component was 0.302, which was not significant.

For PC3 and PC4, the mutated simulation data seems to spend most of its time in roughly three different conformations while the wild type data is more concentrated in two. The mutated simulations have more data that falls high on the PC3 axis, meaning that the complex became more "twisted" in these simulations. There was also a higher variance in the data distribution along PC3 in these simulations. However, the wild type simulations had a higher variance in the data distribution along PC4 when compared to the mutated simulation data; as the wild type complex samples structures with very low PC4 values (around –250). This is likely the conformation where the tail of EloB interacts with A3F. Even though the wild type simulations sample this conformation, in general, it appears that the mutated simulations spend more time in conformations reflected by lower PC4 values. These conformations would have the tail of EloB more extended and, the complex in a slightly more linear orientation, and a different degree of twist for the A3F protein. The p-values for the differences in data variance along PC3 and PC4 were 0.587 and 0.079, respectively.

In general, the mutated simulation data appears to be more similar to the starting structure in the motions captured by PC1 and PC2 (36.1% of the total data variance) and becomes less folded and closed. The mutated simulation data also has a higher spread of data along the PC3 vector, meaning these simulations experienced a higher range of twist. The wild type simulations sampled a conformation in PC4 characterized by the interaction of A3F and part of EloB, and also experience a higher range of A3F twist. However, the mutated complex data spends more simulation time in a conformation that is farther from the starting structure in this dimension. Though these differences were observable from the projection of the simulation data onto the top four PCs, they were often subtle and never statistically significant, with the smallest p-value being 0.079. However, in order to get a better understanding of the differences between the data distribution along the top four principal components for these two simulation types, I used Scikit-Learn's k-means clustering algorithm on the data.

In order to determine the optimal number of clusters to use, I scaled the data and plotted the inertia of the k-means for various cluster values and looked for the "elbow" (See Figure 9); the first four PCs were included[30]. (I also tried other clustering algorithms, such as DBSCAN, but found k-means worked best.) The figure produced by this process varied each time, and I was not able to determine a perfect "elbow", but after considering various options and plots, I decided to use five clusters. The five clusters determined by k-means can be seen in Figure 10 (the data from both the mutated simulations and wild type simulations are plotted together to help showcase the structure of the clusters). In order to better understand the differences between these five clusters, I determined the "average structures" of each of the clusters. To do this, I found the mean of the x, y, z coordinates for each atom among all the frames that fell into each cluster using the NumPy mean method. After I found the average structure for each cluster, I calculated the RMSD between this average structure and all of the trajectory frames that fell into each respective cluster, for the wild type simulations and the mutated simulations. I did this

because the calculated average structure is a structure that does not actually exist, and I wanted to be able to visualize a real structure from the simulation data. So, I found the frame number for each simulation type which was closest to the average structure and chose the frame with the smallest RMSD (between the smallest from the wild type and the smallest from the mutated simulations). I was then able to visualize these frames in VMD (Figure 10).

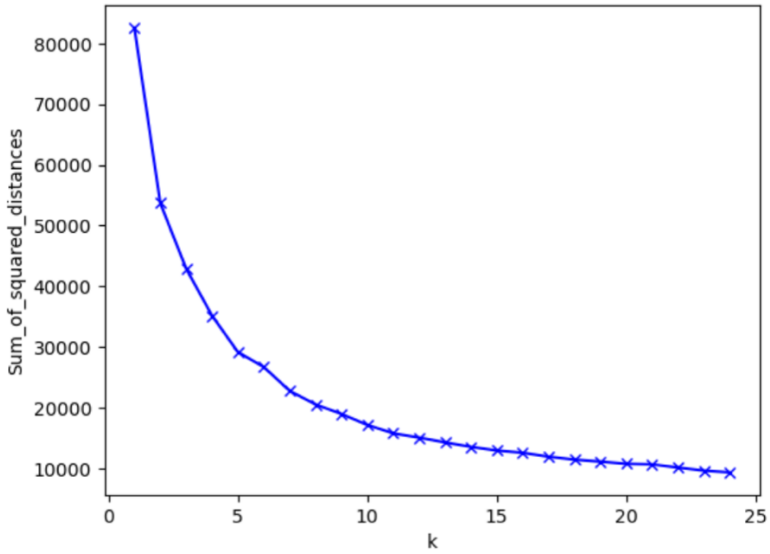


Figure 9. The elbow method to determine the optimal number of clusters to use for k-means clustering. The first four PCs were included in the data considered and a k of five was chosen.

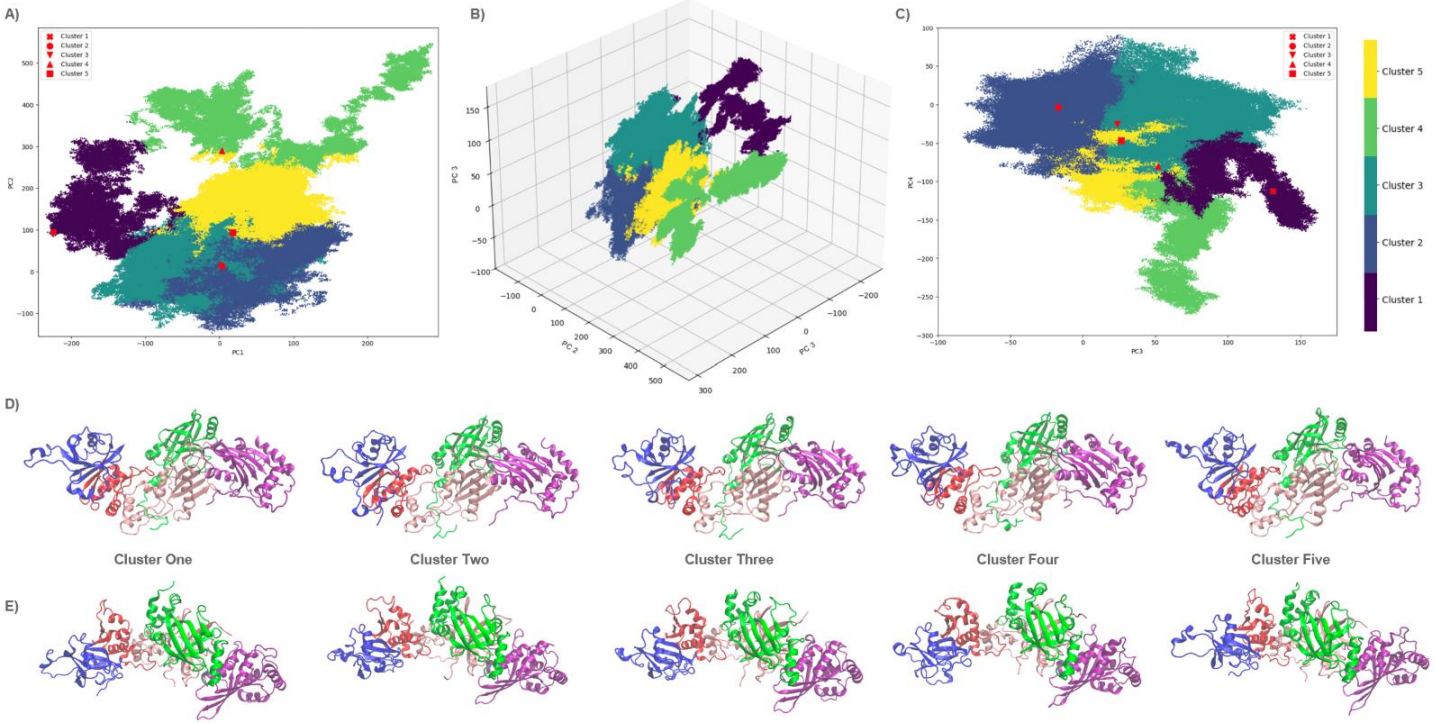


Figure 10. The five clusters determined by k-means clustering projected onto A) the first two PCs and B) the first three PCs, and C) the third and fourth PCs. The first four principal components were included in the clustering and the shown data is from both the wild type and mutated simulations. The red shapes showcase the average structure for each cluster; these structures are visualized from D) the side view and E) the top view.

When the mutated simulation data and the wild type simulation data are projected separately onto the top PCs with the cluster coloring, their differences become more obvious (Figure 11). One of the most striking differences is that nearly none of the wild type simulation data falls into cluster one; only 96 data frames out of the 480,000 total frames fall into this cluster. The average structure for this cluster (like that of cluster five), has a more outward reaching EloB, visible mainly from the top-down view (Figure 10). Additionally, while both simulations have data that falls into cluster four, the distribution and shape of this data is noticeable. Mainly, the wild type simulation data has more data falling into this cluster (57,391 points as compared to 16,248 points) and samples a larger amount of the cluster (the data is more spread out along all four of the top PCs). The average structure for this cluster might not very representative of the whole cluster as the distribution of data is very spread out in noticeable patterns, but the average structure does appear to be more "open" and "folded" than its neighboring cluster five structure. From the top, this cluster also appears to have A3F twisted in an orientation that makes it stick out farther from the complex. The population of clusters two, three, and five are more similar between the two simulations, though the sizes of the clusters differ slightly. The largest difference is cluster five with 122,824 points in the wild type simulations and 68,190 in the mutated simulations; the average structure for this complex is more linear, folded, and EloB and EloC are twisted in a different orientation than in all the other average structures.

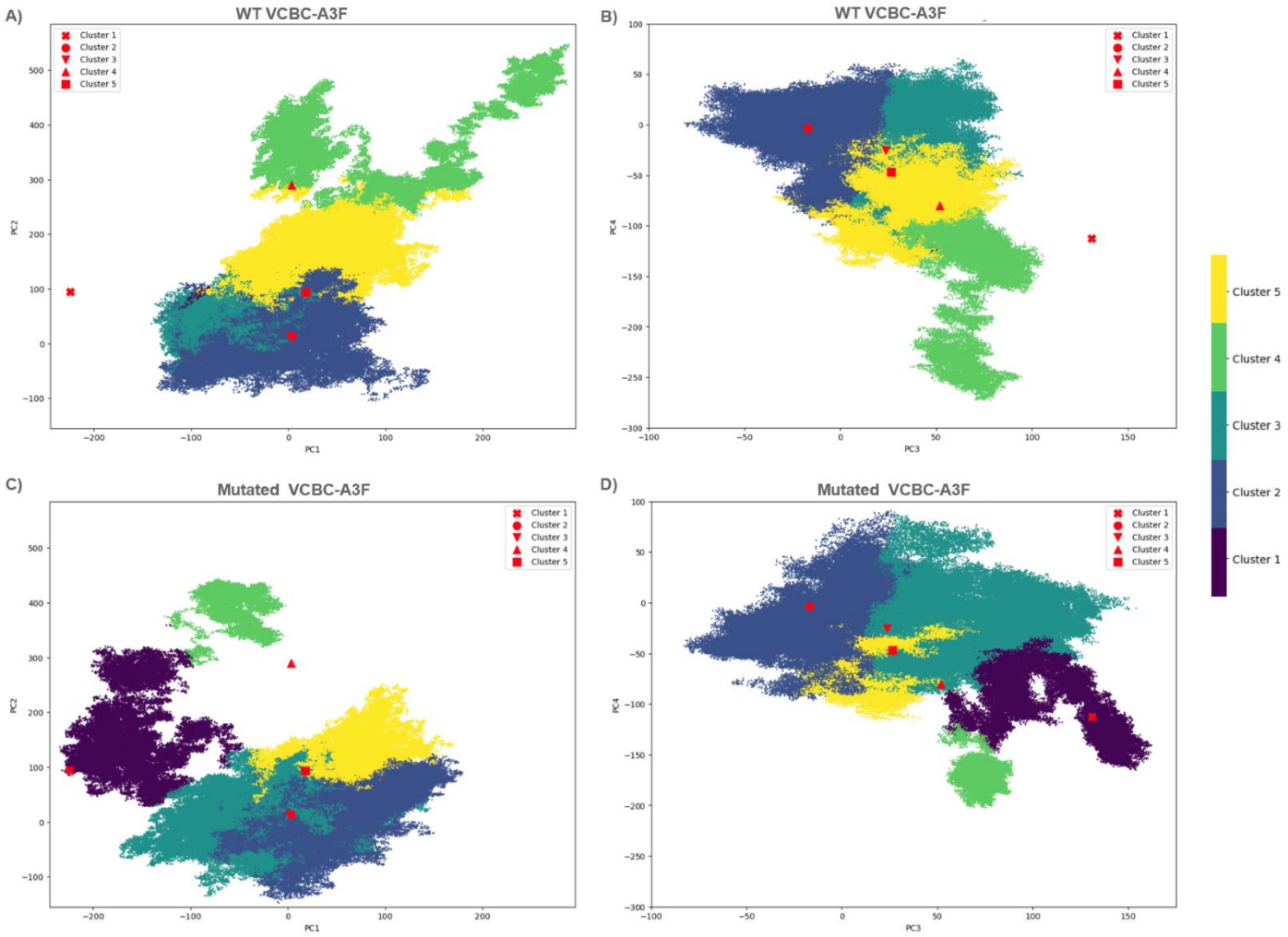


Figure 11. A) The wild type simulation data projected onto the first two PCs and the B) third and fourth PCs with clusters colored. C) The mutated simulation data projected onto the first two PCs and the D) third and fourth PCs with clusters colored.

Overall, the use of PCA and k-means clustering is able to highlight some differences between the wild type and mutated simulations. PCA isolated the top four principal components which accounted for 55% of the total data variance (synonymous with global motions) of the complex, and by plotting the distribution of simulation data onto these top four principal components, I was able to determine that the mutated complex appears to be less folded and closed throughout the simulations and may also have less interactions between the tail of EloB and A3F. After using k-means clustering to determine the five different clusters that the data falls into using the top four principal components, I was also able to determine that the wild type simulation data rarely falls into cluster one, characterized by a more outward reaching EloB. While these differences were interesting, later we became skeptical of the validity of these results. The unexpected standard deviation values at first did not raise concern due to their

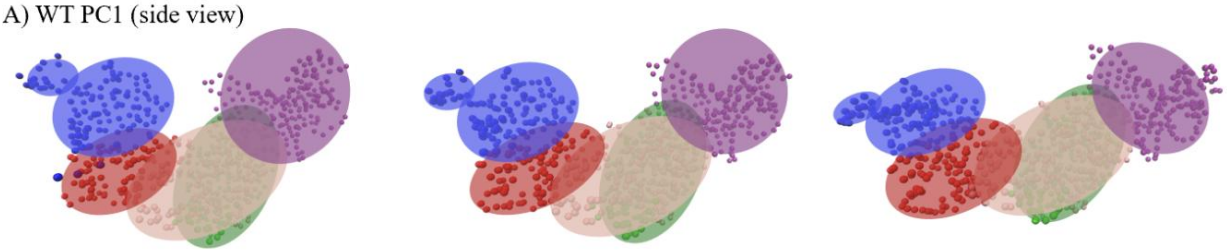
unexpected orders because the principal components were determined by the combined data (Table 2). However, the next chapter further reinforces the idea that something might be wrong with these results. Regardless, this chapter still highlights how PCA can be used to help visualize MD simulations and isolate important features or differences between simulations.

5. Further Exploration of PTRAJ PCA

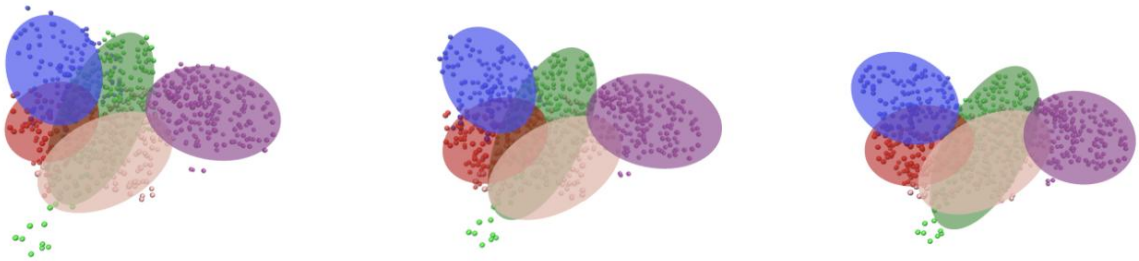
The previous chapter calculated the top ten principal components using the combined data from the wild type and mutated simulation data. This is the standard we have used when looking at the principal components of differing simulations in order to find eigenvectors that are meaningful for the complex's motions across all relevant simulations that will be projected onto them. However, we thought it might also be interesting to calculate the principal components using exclusively the wild type simulation data or the mutated simulation data to see if 1) the principal components differed significantly and 2) when projected onto each other, if they experienced similar variance in their data. To do this, I continued to use PTRAJ to carry out the steps described in the previous chapter, though when calculating the principal components, I loaded in either the wild type data or the mutated simulation data, but not both together.

When I used just the wild type simulation data to determine the top ten PCs, I found that 63.30% of the total variance in these simulations were accounted for in the top four principal components (Table 3). When I visualized these four principal components, I found that PC1 was, again, a type of clamshell motion: the complex started with A3F and EloB closer to each other and along the PC, these move farther apart to a more linear conformation with Vif acting as the hinge point of this motion (Figure 12). PC2 was a highly confusing motion; the complex was almost entirely folded in on itself with A3F and EloB nearly touching, though moving farther apart along the PC. The visualization of this principal component is not reflective of actual conformations sampled by the complex during simulation time, but is still the vector of second highest motion variance. (Since, we later noticed potential flaws with this method, as described in the next chapter, this might be a result of inaccurate projections along PC2 resulting in unrealistic limits of the motion.) PC3 is another type of twisting motion, with the complex staying (relatively) stationary apart from EloB and EloC which twist together. Lastly, PC4 is characterized by more subtle motions. All the proteins are rotating and moving in various ways, but there are no obvious twists or large changes, though the complex does seem to stretch out and straighten out over the duration of the PC.

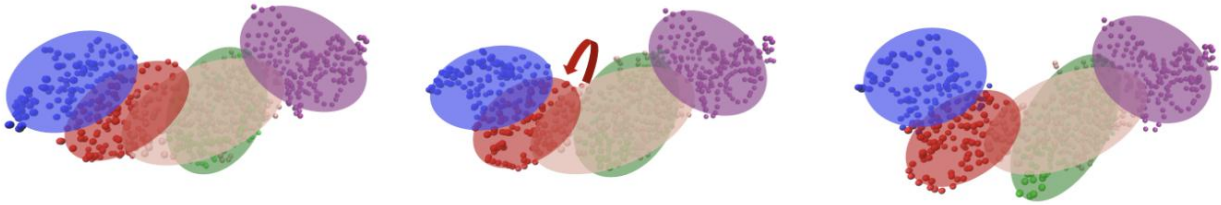
A) WT PC1 (side view)



B) WT PC2 (side view)



C) WT PC3 (side view)



D) WT PC4 (side view)

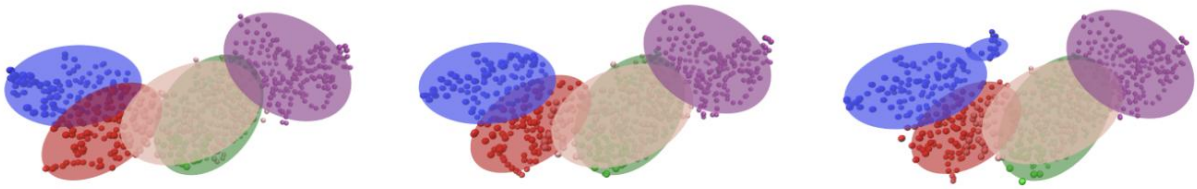


Figure 12. Visualizations of the top four wild type PCs from left to right; videos showing the movements can be seen here; the proteins from left to right are (blue) EloB, (red) EloC, (green) CBF-ß, (pink) Vif, and (purple) A3F. A) PC1 starts in a more "closed" conformation with EloB closer to A3F and ends in a more "open" conformation. B) This visualization of PC2 is likely invalid as the proteins are on top of each other in a way that is not seen throughout the duration of simulation time, though EloB and A3F do move away from each other slightly along this PC. C) PC3 entails EloB and EloC twisting together as the rest of the complex remains relatively stationary. D) PC4 does not have many large, noticeable conformation changes, but the intrinsically disordered region in EloB does move significantly.

Table 3. Percentages of the total data variation captured in the top seven wild type eigenvectors

Principle component	Percentage of total variance	Cumulative percentage
1	23.44%	23.44%
2	19.73%	43.17%
3	12.43%	55.60%
4	7.71%	63.30%
5	4.66%	67.97%
6	3.52%	71.49%
7	3.07%	74.56%

After projecting the wild type and mutated simulation data onto the top four wild type principal components, there were a few differences I was able to observe (Figure 13). The wild type simulation data appeared slightly more spread out than the mutated simulation data along all four principal components, though this difference was less noticeable than expected. Along principle component two especially, the mutated simulation data had less variance. Though I was expecting the wild type data to have higher standard deviations along all four principal components (since these components were meant to represent the four axis of highest data variance for these simulations), this was not what I observed. Along PC one, the mutated simulation data had a higher standard deviation, which was especially surprising. However, more surprising was that the standard deviation of the second principal component for the wild type data was higher than that of the first principal component. If these principal components are being calculated correctly, I would expect the wild type simulation data to have decreasing standard deviations correlated with lower principal component numbers.

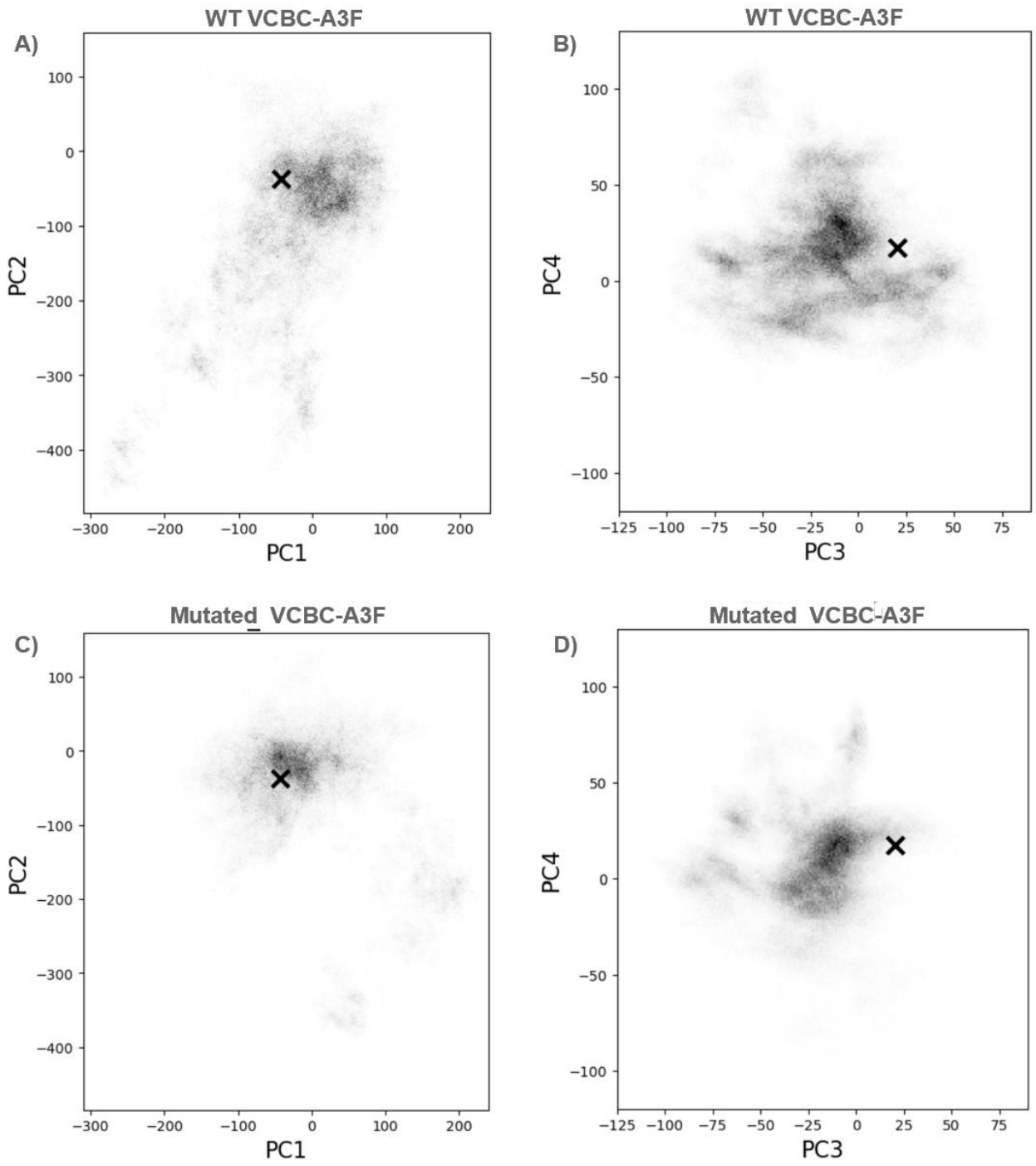


Figure 13. A) The wild type simulation data projected onto the first two WT PCs and the B) third and fourth WT PCs. C) The mutated simulation data projected onto the first two WT PCs and the D) third and fourth WT PCs. The black X represents the starting structure for the simulations; the darker areas in the plots correlate to more simulation time spent in that conformation and lighter areas were sampled less in the simulations.

Table 4. Standard deviations and means of simulation data along the top four wild type PCs.

		PC1	PC2	PC3	PC4
Standard Deviation	Wild Type	69.36	104.36	31.06	27.86
	Mutated	76.54	83.84	26.30	26.44
Mean	Wild Type	-19.22	-99.49	-15.70	12.87
	Mutated	-8.11	-53.28	-23.44	7.77

I noticed a similar, unexpected result when I followed the same procedure using the mutated simulations to find the top ten principal components. In this PCA, the top four PCs accounted for 58.83% of the data variance (Table 5). The first principal component starts with a more folded complex with A3F and EloB closer to each other and straightens out to a more linear conformation (similar to that of PC1 from the wild type principal components) (Figure 14). The second principal component is characterized by A3F subtly rotating into Vif and EloB and EloC twisting slightly and moving away from the complex. PC3, similarly to the wild type PC2, is another questionable PC with the complex starting off unrealistically compacted and expanding. Again, the vector of this motion could be correct, but the fact that the proteins are overlapping at the extreme end of the PC implies there is something wrong with either the motion or the data projection. PC4 has the majority of Vif almost entirely stationary while A3F rotates at the binding site and the tail of EloB expands out towards A3F.

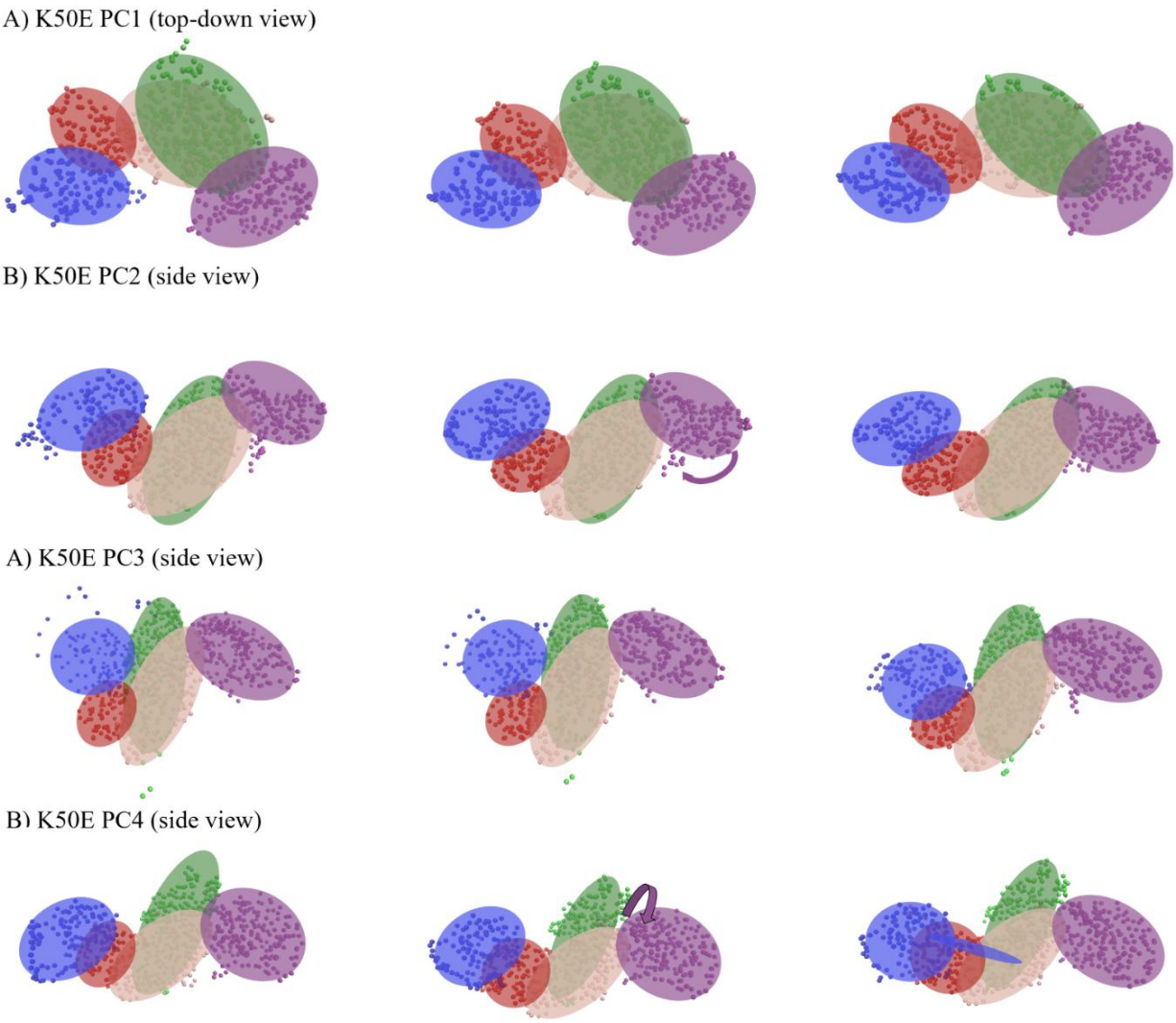


Figure 14. Visualizations of the top four mutation PCs from left to right; videos showing the movements can be seen here; the proteins from left to right are (blue) EloB, (red) EloC, (green) CBF-ß, (pink) Vif, and (purple) A3F. A) PC1 starts in a more "folded" conformation with EloB closer to A3F and ends in a more "open" conformation. B) In PC2, A3F subtly rotates towards Vif and EloB and EloC move slightly farther away from the rest of the complex. C) This visualization of PC3 is likely invalid as the proteins are on top of each other in a way that is not seen throughout the duration of simulation time, though EloB and A3F do move away from each other slightly along this PC. D) PC4 is characterized by the rotation of A3F at the Vif-A3F binding site and the movement of the EloB tail towards A3F.

Table 5. Percentages of the total data variation captured in the top seven mutation eigenvectors

Principle Component	Percentage of total variance	Cumulative percentage
1	21.47%	21.47%
2	16.40%	37.87%
3	12.84%	50.71%
4	8.12%	58.84%
5	4.75%	63.57%
6	4.59%	68.15%
7	3.55%	71.70%

After projecting both simulation data types onto these principal components, I was once again able to observe some differences (Figure 15). In these two-dimensional projections, the mutated simulation data appears to be more spread out in all directions (forming a roughly circular shape) compared to the wild type data (which looks similar to a brushstroke). Despite this, the wild type simulations actually have larger standard deviations along all the top three principal components (Table 6). Though it is slightly harder to tell from these plots compared to the previous ones, it looks like the mutated simulation data could also be forming more/different clusters along these four principal components.

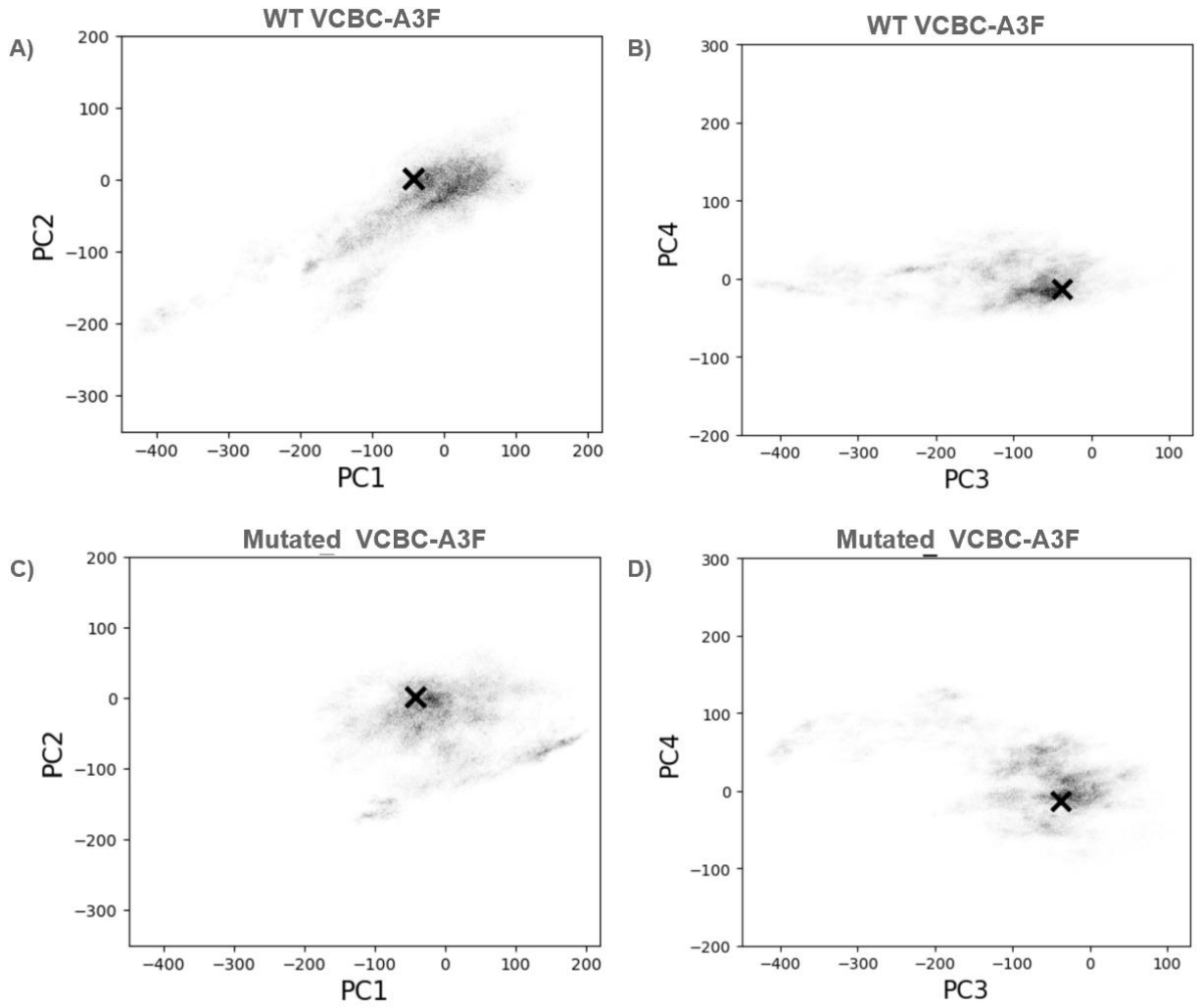


Figure 15. A) The wild type simulation data projected onto the first two mutation PCs and the B) third and fourth mutation PCs. C) The mutated simulation data projected onto the first two mutation PCs and the D) third and fourth mutation PCs. The black X represents the starting structure for the simulations; the darker areas in the plots correlate to more simulation time spent in that conformation and lighter areas were sampled less in the simulations.

Table 6. Standard deviations and means of simulation data along the top four mutation PCs.

		PC1	PC2	PC3	PC4
Standard Deviation	Wild Type	90.81	57.01	92.95	22.69
	Mutated	73.99	46.37	86.87	40.84
Mean	Wild Type	-37.18	-34.61	-112.25	-2.37
	Mutated	-5.81	-33.89	-69.48	14.24

Despite any potentially interesting differences observable from the data projections, the standard deviations again made me skeptical of these results. The wild type standard deviations were higher than the mutated standard deviations for the top three principal components; while this could just mean that the wild type complex was significantly more flexible than the mutated complex, it does not make sense that the wild type standard deviation for these principal components are sometimes higher than the wild type standard deviations when the principal components were calculated using the wild type data. It also does not make sense that the standard deviation from the mutated simulations projected onto PC3 is higher than that from PC1 or PC2.

These results, especially regarding the unexpected orders of the standard deviation values, led me to believe something in our methodology was flawed. Unfortunately, I did not have time to identify what could be causing this problem. There is likely either something wrong with the PTRAJ commands we have been using to calculate the principal components or the projection of the data onto the found principal components. If both the principal components were being calculated correctly and that data was being projected properly, the standard deviations should be in decreasing order. This was not a problem we had noticed in the past since, like in the previous chapter, we normally run PCA analysis on the combined data. In this case, the standard deviations may not be strictly decreasing for the wild type data or the mutated data since the PCs were calculated on the combination of the two. However, running PCA on each of the simulation types separately illuminated a problem in our methodology. In the future, it would be interesting to identify what the problem is and rerun this analysis on more equilibrated simulations in order to get a better understanding of the global motions of the complex before and after the mutation.

6. Visualizing Complex Motions using Scikit-Learn PCA

Using PTRAJ software to calculate the PCA of MD simulation data is useful as it provides the functionality of visualizing the principal components (i.e. Figure 7). However, there are a variety of well-established PCA methods. After noticing the unexpected standard deviations from the previous chapter, I decided to explore another PCA method to compare my results and help determine the validity of the PTRAJ PCA as well as provide insight to the ordering of the standard deviations. I used the Python Scikit-Learn's library's PCA functionality in order to create a standard for comparison.

To do this, I loaded in the data using mdtraj (every tenth frame, like from before), removed all the elements that were not part of the backbone, and aligned the structures to the first frame from the mutated simulations. I originally used StandardScalar to scale the data, but we decided this was causing disproportionate differences in the coordinates, so I removed it in the final versions. After I ran the PCA, I used the built in PCA attributes to determine that 53.14% of the total data variance fell within the top four calculated principal components, which is not as high as would be desired for analysis (Table 7). Unfortunately, due to the nature of this method, I also was not able to characterize what these principal components represent structurally. In the future, it might be interesting to investigate if it's possible to convert the principal components found in Python to a form that's interpretable by PTRAJ or can be converted to an mdcrd file that's viewable in VMD.

Table 7. Percentages of the total data variation captured in the top seven eigenvectors

Principle Component	Percentage of total variance	Cumulative percentage
1	20.39%	20.39%
2	14.47%	34.85%
3	11.65%	46.51%
4	6.63%	53.14%
5	5.35%	58.49%
6	3.88%	62.38%
7	3.49%	65.87%

When I projected the simulation data onto these top four principal components, they did look different compared to the PTRAJ PCA, mainly in the fact that the data was more tightly centered (Figure 16). In general, the wild type simulation data was more spread out along PC1 and PC2 (though the difference in the standard deviations along these principal components were not significant with p-values of 0.736 and 0.188 respectively). Alternatively, the mutated simulation data was slightly more spread out along PC3 and PC4 (though again, not significantly different with p-values of 0.786 and 0.590 respectively). More interesting than the standard deviations of the data along each principal component, perhaps, are the clusters formed by the

data. The wild type data seems to sample two different clusters along PC1 and PC2 whereas the mutated simulation data spends all simulation time in one cluster (Figure 16 A, C). Similarly, along PC3 and PC4, the wild type data appears to sample roughly three different clusters while the mutated simulation data samples 2-3, with a single cluster falling separate from the rest of the data, higher up along the PC3 axis. It is also interesting how far the data is from the starting structure, implying the ensemble sampled does not overlap with the starting structure.

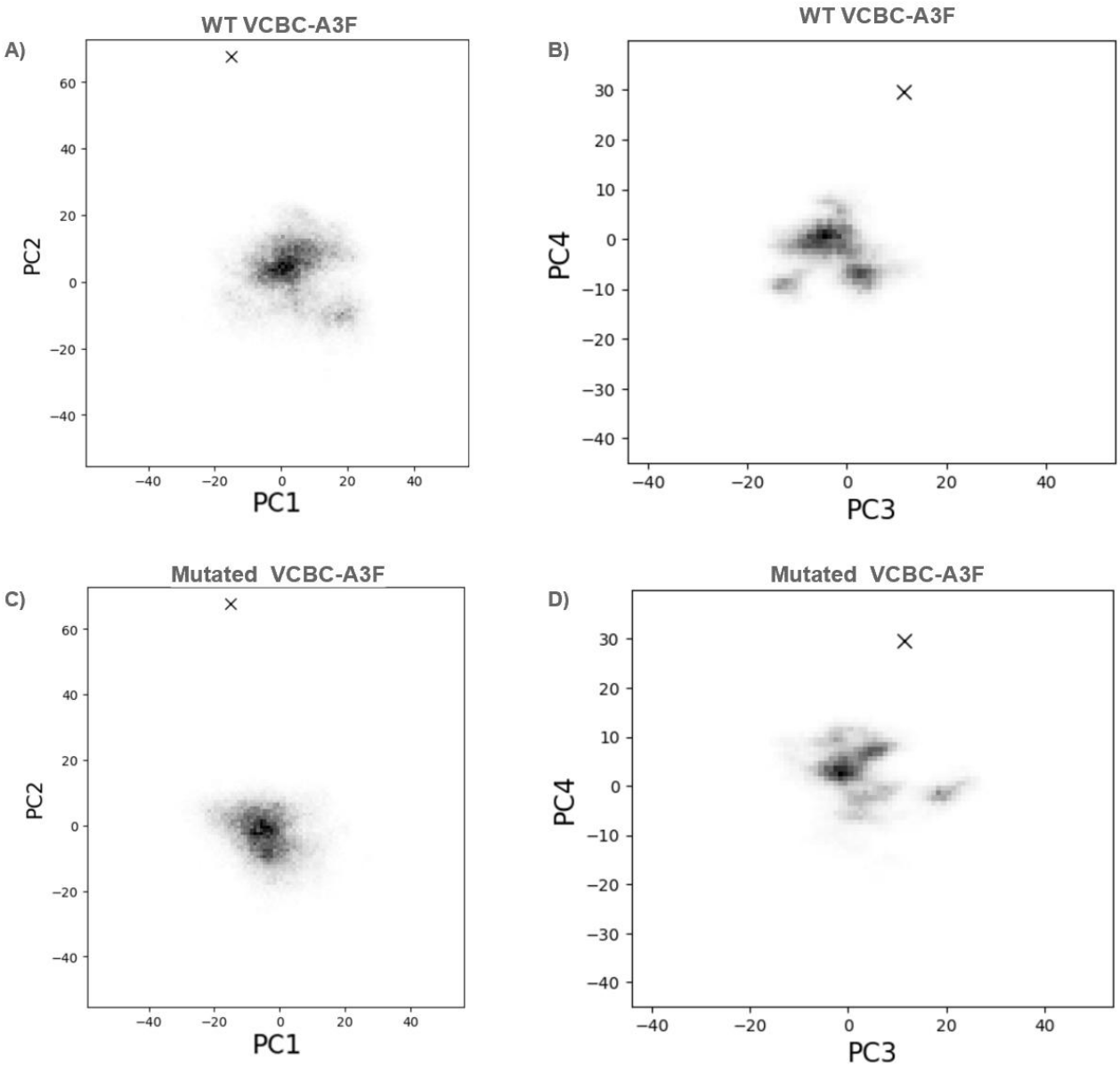


Figure 16. A) The wild type simulation data projected onto the first two PCs and the B) third and fourth PCs. C) The mutated simulation data projected onto the first two PCs and the D) third and fourth PCs. The black X represents the starting structure for the simulations; the darker areas in the plots correlate to more simulation time spent in that conformation and lighter areas were sampled less in

the simulations. Compared to the PTRAJ PC's (from Figures 8, 13, and 15), these PC values fall on a much narrower scale, indicating a difference in how they are calculated.

Table 8. Standard deviations and means of simulation data along the top four mutation PCs.

		PC1	PC2	PC3	PC4
Standard Deviation	Wild Type	9.80	8.84	6.20	4.80
	Mutated	8.41	6.99	7.70	5.46
	Combined	9.98	8.40	7.54	5.69
Mean	Wild Type	4.01	2.67	-2.82	-2.44
	Mutated	-4.01	-2.67	2.82	2.44
	Combined	3.11e-05	-4.85e-06	4.66e-05	-4.92e-05
P-value Difference in Variance		0.736	0.188	0.786	0.590

**The standard deviations for this PCA method make more sense as, in the combined data, they are strictly decreasing with each principal component, and for each simulation type, they are still steadily decreasing for the most part, though PC3 for the mutated simulation data is an exception that is most likely a result of using combined data to calculate the PCs. However, I do find it interesting that the means of the PCs are equivalent to the negation of each other, with the combined data's mean being approximately zero for each principal component. This could be a result of how Scikit-Learn's PCA is calculated, though it might be worth looking into further to ensure it's not another error.*

While this "clustering" can be observed visually, in the future it would be interesting to apply a clustering algorithm to this data spread, like that described in chapter four. One of the main shortcomings of using Scikit-Learn for MD simulation PCA is that we are unable to visualize what the motions reflected by the principal components are, and thus it is challenging to ascribe structural meaning to these or quantify how much motion is captured in each principal component. However, in the future, I think this could be inferred by following a similar procedure as described in chapter five. If a clustering algorithm were applied to this data, it could identify simulation frame numbers that are reflective of the clusters, and those could be visualized to better understand the complex's structure.

Since this version of PCA appeared to be working but was less informative than the PTRAJ PCA, I once again decided to calculate the principal components using first just the wild type simulation data and then just the glutamate simulation data to see if this could potentially help highlight any difference between the two. I followed the same procedure as described above, but did not combine the simulation data before running PCA. The wild type PCA accounted for 61.42% of the total variance in the top four principal components, and when I projected the simulation data onto these four principal components, I was able to see that the wild type data was slightly more spread out along all PCs compared to the mutated data (Figure 17). While the mutated data generally stays condensed to one cluster in these plots (maybe with a

slight exception along PC3), the wild type data appears to form various small clusters and also has larger standard deviations (Table 10). The difference between the wild type and mutation simulation data along PC2 can also be considered significantly different with a p-value of 0.042 (the rest are 0.81 for PC1, 0.31 for PC3, and 0.25 for PC4). Again, while observing these differences indicate to me that, along these principal components, the wild type data is sampling conformations that the mutated simulation data is not, this is hard to quantify or understand structurally.

Table 9. Percentages of the total data variation captured in the top seven wild type eigenvectors

Principle component	Percentage of total variance	Cumulative percentage
1	22.72%	22.72%
2	19.17%	41.89%
3	12.02%	53.91%
4	7.50%	61.42%
5	4.52%	65.94%
6	3.35%	69.29%
7	2.98%	72.27%

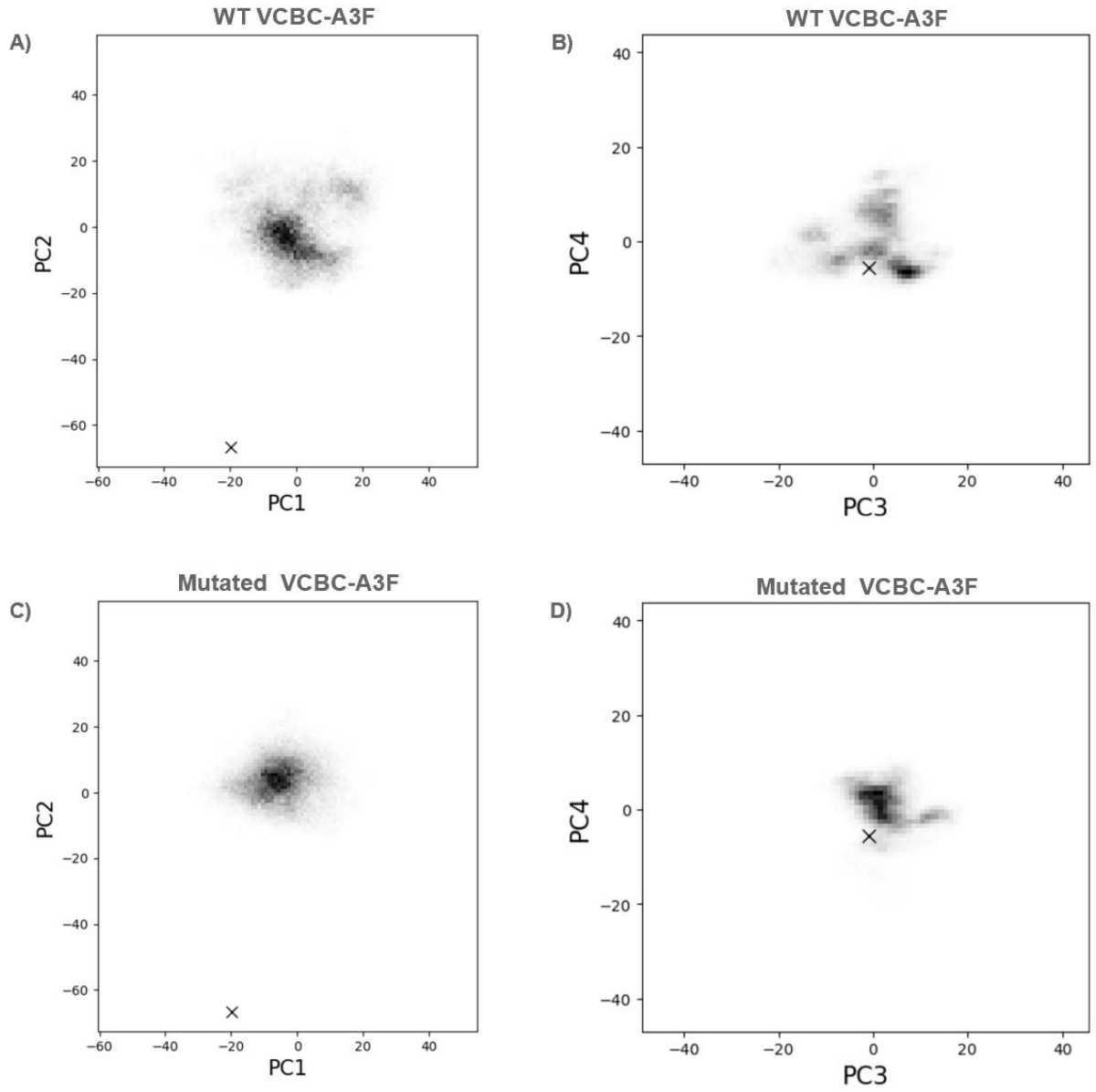


Figure 17. A) The wild type simulation data projected onto the first two wild type PCs and the B) third and fourth wild type PCs. C) The mutated simulation data projected onto the first two wild type PCs and the D) third and fourth wild type PCs. The black X represents the starting structure for the simulations; the darker areas in the plots correlate to more simulation time spent in that conformation and lighter areas were sampled less in the simulations.

Table 10. Standard deviations and means of simulation data along the top four wild type PCs.

		PC1	PC2	PC3	PC4
Standard Deviation	Wild Type	10.20	9.36	7.42	5.86
	Mutated	8.35	6.20	5.09	4.45

Mean	Wild Type	5.80e-05	-4.76e-05	-2.12e-05	- 4.85e-06
	Mutated	-5.14	3.40	2.72	0.44
P-value Difference in Variance		0.81	0.042	0.31	0.25

In the PCA ran using just the mutated simulation data, the top four principal components accounted for 57.01% of the total mutation data variance. From just the projections of the data onto these principal components alone, I was not able to observe many interesting or significant differences between the simulation types (Figure 18). I can see some clusters forming in the mutated simulation data that do not appear in the wild type data, but in the future a clustering algorithm could help determine if these clusters are ones that are only sampled by the mutation data, and visualizing these clusters in VMD could provide insight to what that means structurally. In terms of standard deviations, there was the largest difference between the two along PC2 and PC4 though the no variation along any principal component was statistically significant (the p-value in order from PC1 to PC4 were 0.95, 0.77, 0.46, 0.69).

Table 11. Percentages of the total data variation captured in the top seven mutation eigenvectors

Principle component	Percentage of total variance	Cumulative percentage
1	20.78%	20.78%
2	15.83%	36.61%
3	12.43%	49.04%
4	7.96%	57.01%
5	4.56%	61.57%
6	4.38%	65.95%
7	3.43%	69.38%

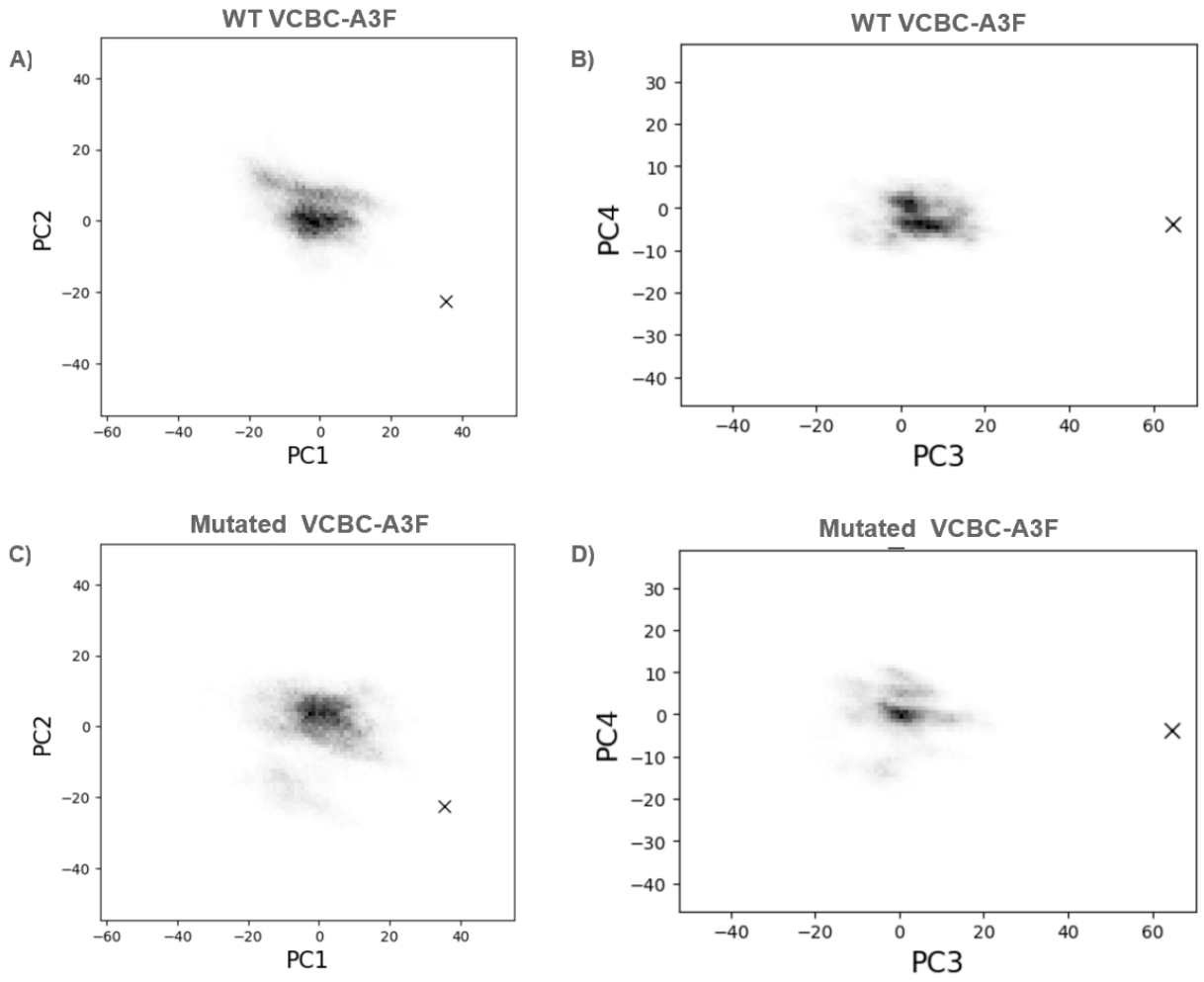


Figure 18. A) The wild type simulation data projected onto the first two mutation PCs and the B) third and fourth mutation PCs. C) The mutated simulation data projected onto the first two mutation PCs and the D) third and fourth mutation PCs. The black X represents the starting structure for the simulations; the darker areas in the plots correlate to more simulation time spent in that conformation and lighter areas were sampled less in the simulations.

Table 12. Standard deviations and means of simulation data along the top four mutation PCs.

		PC1	PC2	PC3	PC4
Standard Deviation	Wild Type	8.57	6.00	7.32	3.81
	Mutated	9.47	8.26	7.32	5.86
Mean	Wild Type	-1.98	3.08	4.53	-2.12
	Mutated	-0.00007	0.00003	0.00007	-0.00002

The Scikit-Learn PCA does appear to work correctly from what I can tell in this chapter; the standard deviations, at least, are in an expected and sensical order. However, from these PCA figures alone, I am not able to draw any definitive conclusions about the structural or conformational differences of the VCBC-A3F complex before and after the K50E mutation. Many of the most interesting differences between the two simulations are the clusters that form in the projections of the data, and without knowing what the principal components represent structurally, it is hard to characterize these in a meaningful way. Additionally, a clustering algorithm should be run to confirm that these clusters are significantly present and distinct. If, in the future, the problem with the PTRAJ PCA cannot be identified, then I believe using Python libraries to run PCA can be a promising analysis method, as long as there is some way to tie the figures back to the complex's structure (which I think would be possible using Python to identify simulation frame numbers to visualize in VMD). Additionally, since the first 300 ns of simulation data were used for this analysis, reanalyzing after 300 more ns of simulation data is acquired would reveal whether further equilibration of the structure leads to more exaggerated differences between the WT and mutant complex.

7. Reducing Dimensionality Through Kernel PCA

As discussed in chapter four, while PCA provides an informative visualization of high dimensional simulation data, it does not work as well when the data does not have strong linear patterns. Figure 19 shows an example of when PCA might fail to identify a noticeable pattern in a spread of data. In order to counteract this potential problem, kernel PCA can be used. Kernel PCA (KPCA) is an extension of PCA which, before running PCA, applies a nonlinear mapping function to the data. This kernel function maps the initial, high dimensional data to a feature space, which is still high dimensional but where the data points might be distributed in a more linear fashion that can be picked up by PCA (Figure 20). The combination of the kernel function and regular PCA provides all the dimensionality reduction benefits of PCA while also allowing nonlinear patterns to be captured. Interestingly, while PCA is a common method used in MD simulation analysis, kernel PCA is much less common, though it has been acknowledged as a potential solution to the nonlinearity of biological systems[31].

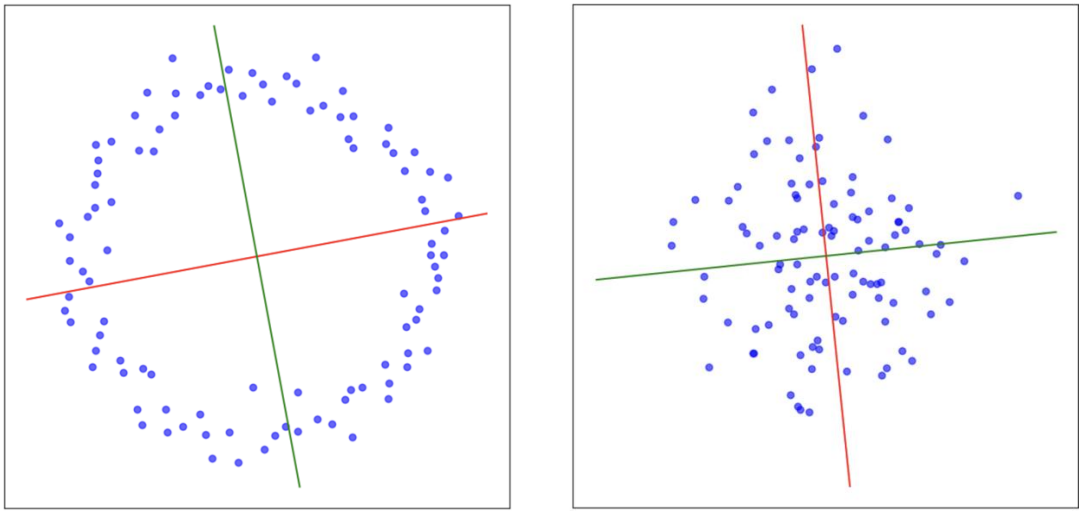


Figure 19. (Copy of Figure 6) Given the two spreads of data shown above, simple PCA will not be able to find meaningful linear patterns in either (the top two principal components for both spreads of data are shown by the blue lines), yet there is a clear pattern in the data on the left.

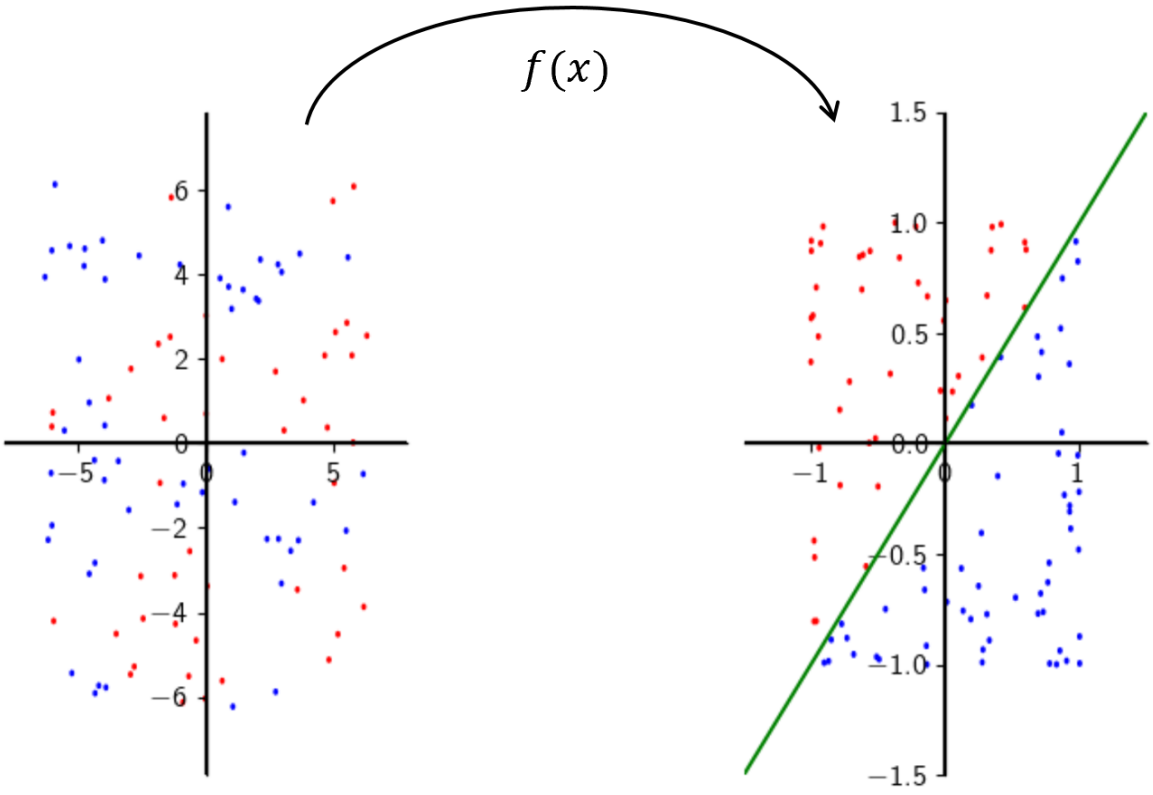


Figure 20: The function f(x) maps the nonlinear data on the left to a feature space where linear patterns can be identified.

To apply KPCA to my simulation data, I used the KernelPCA method from the scikit-learn library[32]. This library includes six different kernel functions which can be used to transform the data before applying PCA. Determining which (if any) kernel function is best suited to a particular dataset is one of the main complications of KPCA, alongside the computational expense. For the purpose of picking out which kernel to use for my simulation data, I tried all of them, though further experimentation of parameters was hindered due to time constraints and computational power limitations. Out of all the built in kernel functions, the redial basis function kernel and the cosine similarity kernel were the only two that were able to consistently run to completion without encountering any errors. To prepare the data for the KPCA, I loaded in every 10th frame from the wild type simulations and the mutated simulations. I then aligned the backbones of the simulations and removed all the atoms that were not a part of the backbone and extracted the x, y, z coordinates of all of the remaining atoms. Unfortunately, due to how computationally expensive and time consuming it is to run this process, I then took only every 25th element from the training data, resulting in only 3,840 frames of data used to train the KPCA. After reshaping the data to NumPy arrays of shape (frame_number, n_atoms*3), I was able to run KPCA.

The radial basis function (RBF) kernel is a commonly used kernel which maps input data to an infinite dimensional feature space by measuring the similarity between data points based on Euclidean distances in the input space[33]. The function can be defined as:

$$k(x, y) = \exp(-\gamma\, d(x, y)^2),$$

where x and y are column vectors, d is the Euclidean distance, and γ is a parameter which controls the smoothness of the decision boundary[34]. I tried four different gamma values with this KPCA; the default of "None" sets gamma to 1/the number of features and I also tried values of one, five, and ten (Figure 21). With the default gamma value, much of the wild type data and the mutated data were overlapping along the first two principal components, however, the wild type data was more concentrated and centered at lower PC1 and PC2 values compared to the mutated data. The center of the wild type data also appeared to be approximately where the starting structure fell onto these PCs. Differences become more noticeable when gamma values were introduced. With gamma values of one, five, and ten, all the wild type simulation data was highly concentrated along both PCs around the location of the starting structure. The majority of the mutated simulation data was also highly concentrated in a similar area, but slightly lower along PC1. Additionally, unlike the wild type data, the mutated simulation data was not entirely restricted to one area. With a gamma value of one, multiple data points experienced high variation strictly along PC1 and a few experienced high variance strictly along PC2. With a gamma value of five and ten, a few data points from the mutated simulations fell in noticeable different locations along PC1 and PC2. These higher gamma values might be giving outliers in the data too much weight, consequently resulting in the majority of simulation data being collapsed into a small area. Regardless, I do think it is interesting that only the mutated simulation data experiences apparent outliers in the PCA figures while the wild type does not.

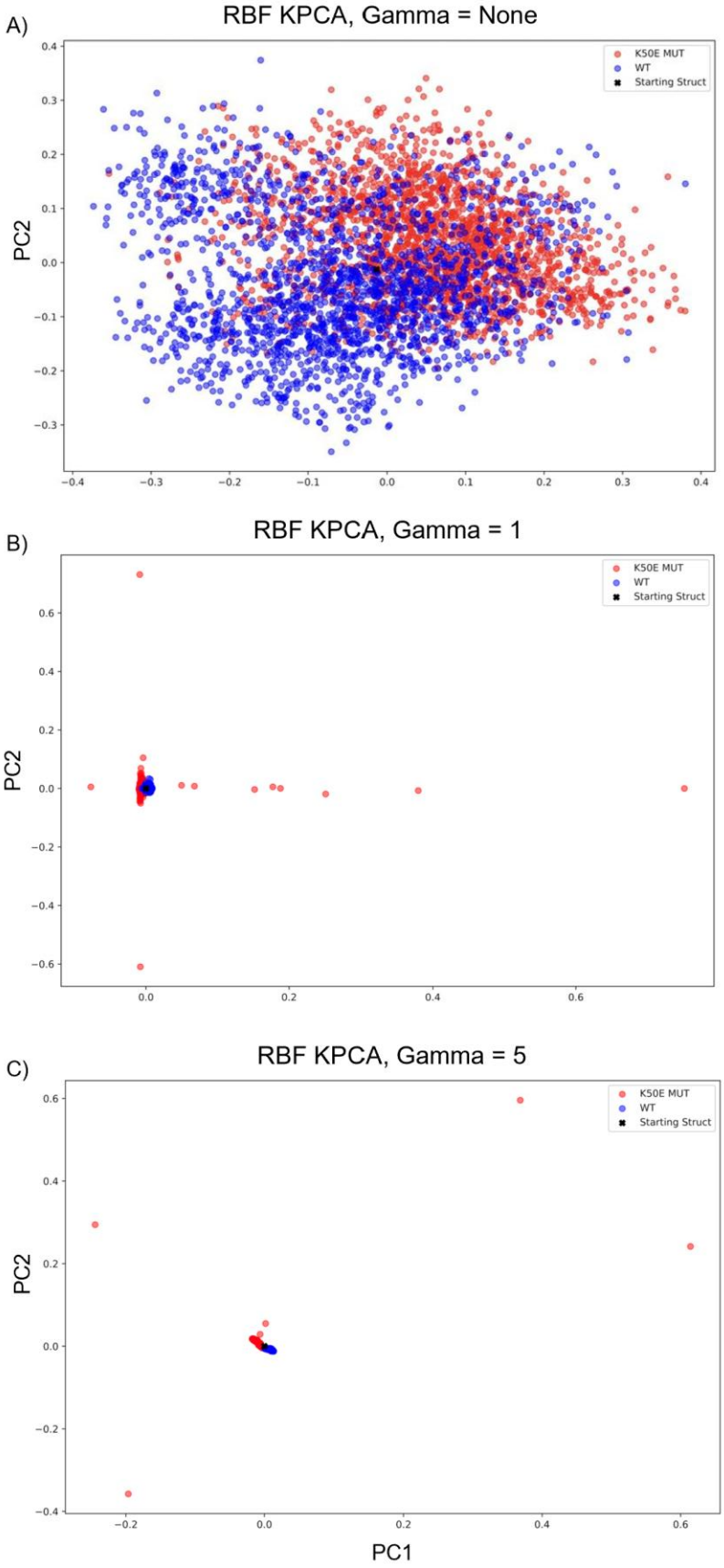

A)
RBF KPCA, Gamma = None
K50E MUT
WT
Starting Struct
PC2
B)
RBF KPCA, Gamma = 1
K50E MUT
WT
Starting Struct
PC2
C)
RBF KPCA, Gamma = 5
K50E MUT
WT
Starting Struct
PC2
PC1

Figure 21. The wild type simulation data (blue) and the mutated simulation data (red) projected onto the first two principal components from the RBF feature space for gamma values of A) none, B) one, and C) five. The plot for the gamma value of ten is not pictured as it was identical to when the gamma value was five.

In order to ensure that the differences in the mutated simulation data and wild type data were not a result of plotting a small fraction of the data, I reran this analysis and plotted all the data points. (The KPCA was still run with only a fraction of the data.) The results were the same—the wild type data was highly concentrated to one area whereas the mutated simulation data experienced some fluctuations along both principal components. This difference, paired with the fact that the data is centered in different spots along both principal components for all gammas indicates to me that these feature spaces had slight differences between the two simulation types that PCA was able to pick up on. Unfortunately, due to time constraints and the nature of this analysis method, I am unable to ascribe meaning to this difference.

I then followed the same procedure with the cosine similarity function. This kernel computes the dot product of the vectors divided by the products of their norms, essentially using the angles between vectors to determine their similarity. If x and y are column vectors, then their cosine similarity (k) is[35]:

$$k(x, y) = \frac{x^T y}{\|x\|\|y\|}.$$

The cosine kernel does not have any gamma value. After training this KPCA model, I projected the wild type and mutated simulation data separately onto the principal components from the feature space (Figure 22). From doing this, I was able to see a similar pattern as that from the RBF KPCA when the gamma was none. Again, the data was mostly overlapping, but with the wild type data centered around a point slightly lower along PC1 and PC2. It is also interesting that these data points fall noticeably far away from the starting structure along PC2. This could be an indication that something is wrong with this methodology, as it seems unlikely for it to be so far away considering its proximity to the data in other figures (i.e. Figure 21). However, it might be possible that this difference is a result of the equilibration process, and therefore, along this principal component, the two different simulations are more similar to each other than the starting structure. If this were the case, mapping the data to a cosine feature space reveals this difference in a way that the RBF kernel did not.

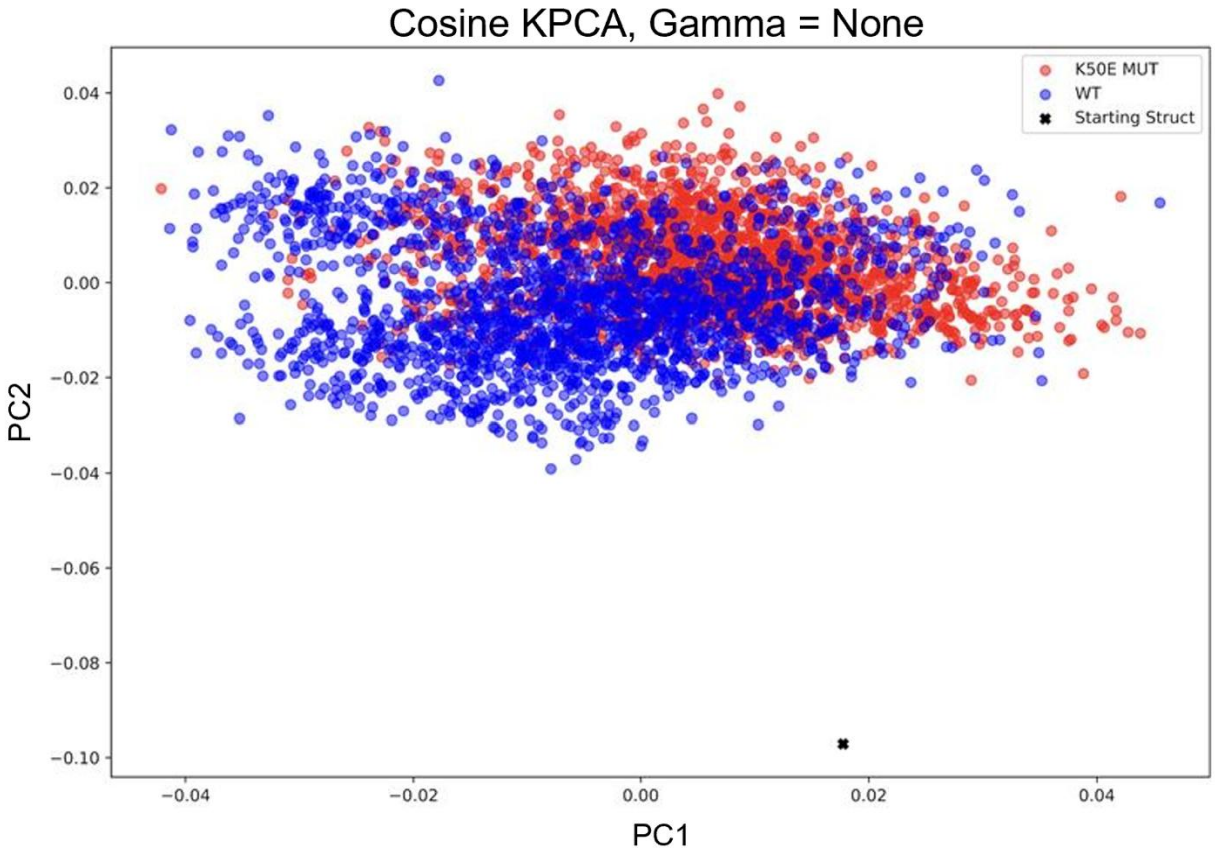


Figure 22. The wild type simulation data (blue) and the mutated simulation data (red) projected onto the first two principal components from the cosine similarity feature space.

Though these kernel functions operate in different ways and create feature spaces based on different metrics, they all effectively are measuring the similarity between data points/vectors. Therefore, if they measure more similarity between the data points from the wild type simulation and less similarity between these data points compared to the mutated simulation data, this difference will be reflected in the feature space in a way that, if it's extreme enough, could be identifiable using PCA. Though at this time, I am unable to quantify or describe the differences identified in the simulation data from the KPCA I was able to run, I think the fact that there were differences (especially in the RBF KPCA with gamma values present) could indicate that there is some structural differences between the wild type and mutant simulation data. I believe that the areas in the KPCA figures where there was no overlap between the two might indicate structural conformations that are sampled by either the wild type data or the mutated simulation data exclusively. While the differences are subtle, and there is a lot of overlap, this could also be partially because the simulations used for analysis included some equilibration time. In the future, this analysis could be rerun using the last 300ns of 600ns simulation data to see if the differences become more apparent.

8. Visualizing the Latent Space Through an Autoencoder

Another method of reducing the dimensionality of data is through a trained autoencoder. Autoencoders are a type of neural network which, given high dimensional input data, learns how to compress the data and then decompress the data in order to best recreate the original[33]. The compression of data is completed by the encoder; the encoder takes the input data and compresses it. The decompression of the data is performed by the decoder which, given encoded data, can recreate the original input data as accurately as possible. The training of the autoencoder occurs as training data is passed through the encoder and decoder; the reconstructed data is compared to the original and the weights in the neural network are updated to punish poor reconstruction (see Figure 23 for a visualization of an autoencoder's structure). Because the dimensionality of the encoded data or "latent space" is so much lower than the original data, the model is able to learn the useful properties of the data as it must prioritize which aspects of the input are important enough to be reflected in the encoded data[33]. In other words, information about the input data which is irrelevant will not be reflected in the latent space, as this information will not be helpful with the reconstruction of the data and there is no room in the latent space to house irrelevant information. While autoencoders need to be trained with an encoder and a decoder, many applied uses of the autoencoders need simply the encoder or decoder alone.

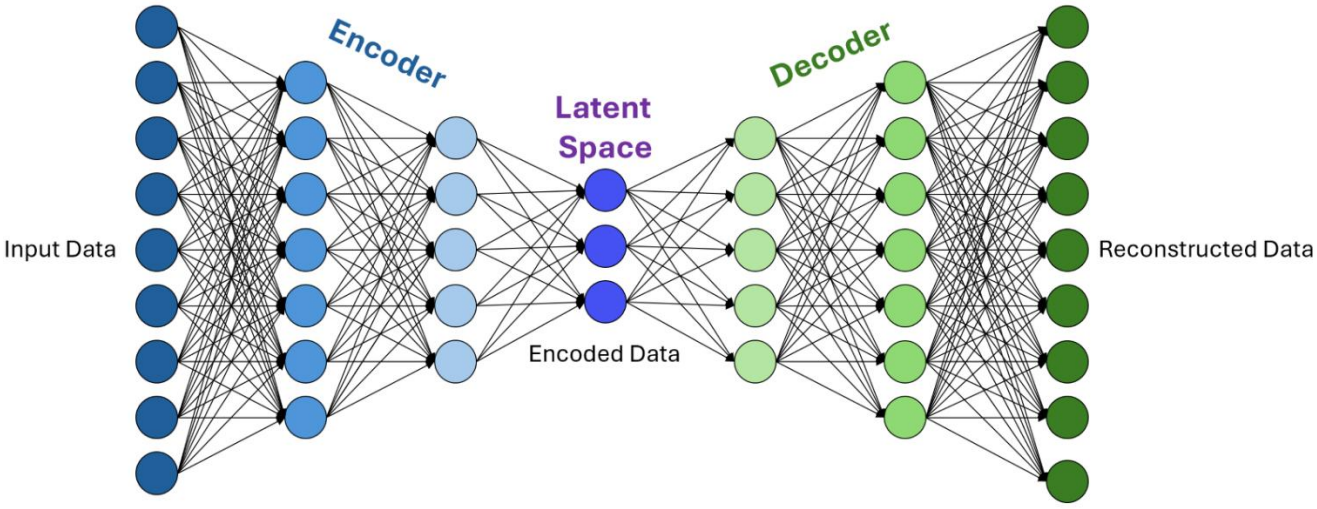


Figure 23: Example structure of an autoencoder. The input data gets depressed in dimensionality to the latent space by the encoder. The encoded data then gets reconstructed by the decoder.

Previous studies have found the decoder useful for generating new data; for MD simulation data specifically, this has mainly come in the form of generating protein ensembles and molecule conformations[13,36]. While this is an exciting capability that is gained from having a

trained autoencoder, this research focuses on the uses of the encoder as a way to reduce the dimensionality of the MD simulations in order to gain a better understanding of the important features of the simulations. Previous research has found that the latent space created after the encoding process through a variety of different autoencoders can help identify major states and important conformational transitions in MD simulations[13]. This is possible because, as the encoder maps data points to the latent space, similar ones aggregate into clusters and these clusters can be characterized using the decoder. This effectively informs us about what different conformations are sampled throughout simulation time. Metteo Degiacomi published one such study and provided their files for creating and training their autoencoder; this acted as a starting point for the structure of my autoencoder, though the implementation and final structure differ in a variety of ways[37].

In order to train the autoencoders, I started with preparing my data; since the autoencoder should be capable of interpreting both the wild type and mutated simulation data, I used data from both sets of simulations to train it (note: only every tenth frame of the simulations were loaded in due to memory constraints). In order to make the data the same dimensions for the two simulation types (there was an initial difference due to the number of atoms changed by the mutation) and to help with training speed and memory consumption, I removed all the atoms that were not a part of the backbone from the trajectory. This decreased the data from a (48,000, 12,053/12,060, 3) array to two (48,000, 2,968, 3) arrays. Additionally, I used the mdtraj library superpose method to align the two simulations in space. I then split both the wild type and mutated simulation data into two pieces: 7/8ths of the data was used for training and 1/8th was used for testing. Testing data/validation data is crucial to prevent the neural network from overfitting to the training data (the network should learn important information in order to handle new data rather than simply memorize the training data). Since there were eight independent simulations for each of the complexes, one simulation from each was retained for validation data (this was opposed to random frames of the data, which might have been close to identical to the training data). This process resulted in training data of dimensions (84,000, 2,968, 3) and testing data of dimensions (12,000, 2,968, 3). The NumPy arrays were then flattened into dimensions that were usable for the autoencoder and scaled using StandardScalar (when the data was not scaled, all data was encoded to a single point at 0,0,0 in the latent space).

The Python library TensorFlow was used to construct and train the autoencoders. A variety of different autoencoder depths, activation functions, learning rates, and epochs were trained and tested in order to fine tune the parameters to best suit the task at hand, though time constraints limited the extent to which different autoencoders could be built and trained. All autoencoders were trained to compress the input data to three dimensions before decoding it and used Adam as the optimizer with root mean squared error (RMSE) as the training metric. The two main hidden layer activation functions that were tested were the Tanh function and the

ReLU function. The Tanh (hyperbolic tangent) function was considered as it is able to handle negative values, defined as:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

The ReLU (rectified linear unit) function was chosen as it is computationally inexpensive and would be likely to train better and faster. The function is defined as:

$$f(x) = \max(0, x).$$

The ReLU function does not go below zero, however, so in the future it might be interesting to look into using Leaky ReLU, which would be able to go slightly below zero. These two functions were also chosen as they have been shown to be effective in previous MD simulation autoencoders[13,37]. The output layer always had a linear activation function to ensure there could be negative numbers in the decoded output. Additionally, dropout was always used.

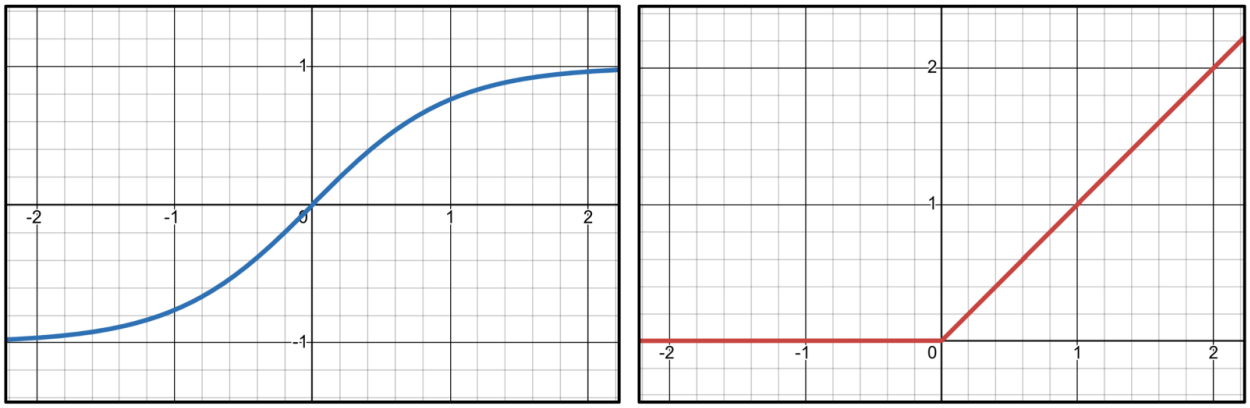


Figure 24: The Tanh function (left) and the ReLU function (right).

For each of these two activation functions, three different neural network structures were tested. The starting structure (aka. the medium sized network) had five dense layers in the encoder and five dense layers in the decoder. This network size resulted in 18,485,947 trainable parameters. I also tried adding more hidden layers with the hopes that a deeper network would be able to learn the complex data better. I added one layer to the encoder and one to the decoder, resulting in a network with 24,560,247 trainable parameters. Since, in earlier experiments I did with this data, shallow neural networks were proving better at learning this data, I also tried training a smaller network with less hidden layers. (These experiments were run before the non-backbone atoms were removed and many deeper networks learned to map all the data to 0, 0, 0 in the latent space.) This network had three layers in the encoder and three in the decoder with 5,370,147 trainable parameters. Given these three different network sizes, the two different activation functions, and learning rates of 0.01 and 0.001, I trained 12 different networks for 100 epochs in order to gain an understanding of which was best. It's important to note that there was a small problem with the protein backbone alignment in these first neural networks. Though this

was fixed in the final networks, it likely strongly impacted the performance of the networks reported below in Table 13.

Table 13. Performances of the initial 12 test networks trained.

Activation Function	Depth	Learning Rate	Mean Reconstruction Error *	Validation RMSE at Epoch100/100
ReLU	Deep	0.001	0.768	0.876
		0.0001	1.174	1.083
	Regular	0.001	0.713	0.844
		0.0001	0.491	0.701
	Shallow	0.001	0.464	0.681
		0.0001	0.235	0.485
Tanh	Deep	0.001	0.316	0.562
		0.0001	0.303	0.551
	Regular	0.001	0.252	0.502
		0.0001	0.273	0.523
	Shallow	0.001	0.278	0.528
		0.0001	0.258	0.508

** The mean reconstruction error was calculated by running the test data through the autoencoder and subtracting the original test data from this reconstructed version of the test data. This was then squared along the first axis, and then the mean was taken to find the reconstruction error for all the test data. The mean of these reconstruction errors is reported above.*

The shallow ReLU network with a learning rate of 0.0001 (in green above) performed best overall, so I decided to train this network for 300 epochs to see if it could improve with more training time. I also chose to train a regular-depth Tanh network for 300 epochs. I did this because I believed the ReLU network performed best because this activation function trains faster and with more training time the Tanh network could eventually perform better. The Tanh network was run with a learning rate of 0.001. If I had more time, I would have liked to train more networks for longer to see how well I could get a network to perform.

Unfortunately, after I fixed the protein alignment and trained the networks longer on the aligned data, they performed worse than the initial networks trained over less epochs (and with improperly aligned data), at least in terms of root mean squared error. The medium-depth Tanh network had a validation root mean squared error of 0.825 by the last epoch and a reconstruction error of 0.681. The shallow ReLU network had a RMSE of 0.863 by the last epoch and a reconstruction error of 0.744. Though this is far from optimal, and in the future, I'd like to try to get these numbers lower, I thought it would still be interesting to see what the latent spaces were from these networks. Despite the relatively poor RMSE of both the networks, even at this performance level they were able to somehow differentiate the wild type data and the mutated

data. The Tanh network plotted the two different simulations almost entirely separately along the Z-axis (Figure 25). This separation was not perfect, but the vast majority of the wild type simulation falls at 1.00 along this dimension compared to the mutated simulation data which falls at –1.00. Because the structure was (properly) aligned and the mutation-specific atoms were removed from the data, this difference must be due to inherent differences in the x, y, z coordinates of the simulation data. Additionally, the crystal structure fell into the mutated simulation data, which might imply that the crystal structure is closer to the mutated simulation data conformations than the wild type conformations.

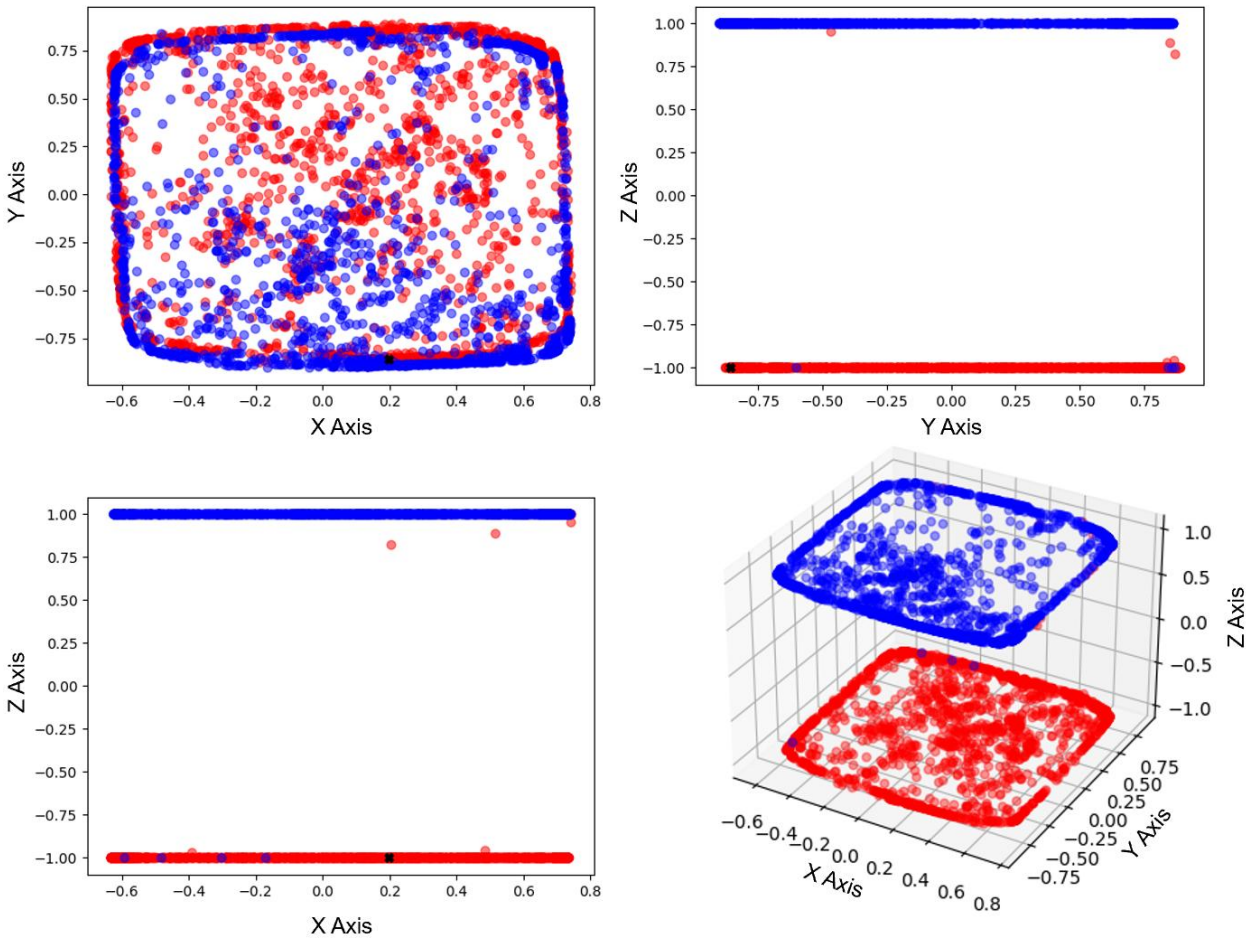


Figure 25: The latent space from the medium sized Tanh network with the encoded wild type data (blue), the encoded mutated data (red), and the encoded starting structure (black X). The amount of simulation data plotted was reduced by taking every 25th data point to improve clarity.

The shallow ReLU network also identified some differences between the two types of simulations. Unlike the previous network, this one did not entirely separate the two types of simulation data as there was significant overlap (Figure 26). However, the mutated simulation

data experienced much more variation along all three of the dimensions when compared to the wild type simulation data. This figure almost appears reminiscent of the RBF KPCA figures from the last chapter and may imply that there is a larger variety in the x, y, z coordinates of the mutated simulation data compared to the wild type simulation data (possibly implying the structure is more flexible or samples more conformations). However, the patterns that the autoencoders are learning cannot be deciphered at this point, so I am unable to draw any concrete conclusions about what is potentially causing the differences observable in the latent spaces.

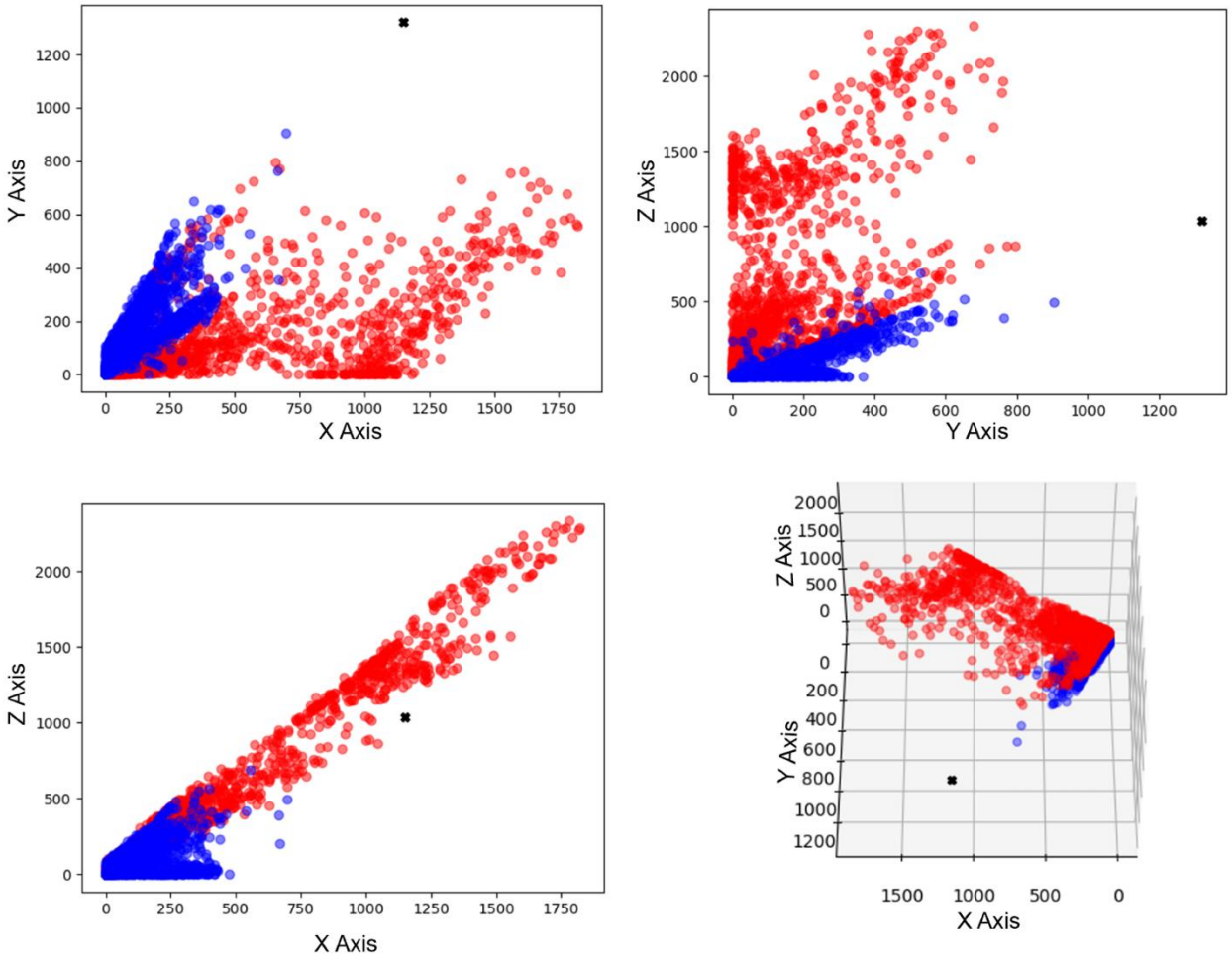


Figure 26: The latent space from the small ReLU network with the encoded wild type data (blue), the encoded mutated data (red), and the encoded starting structure (black X). The amount of simulation data plotted was reduced by taking every 25th data point to improve clarity.

I also decided to try running one more network for a longer period of time just to see if, with more time, it would be able to learn how to better reconstruct this complex data. I decided to train the "deeper" neural network with a Tanh activation function and a learning rate of 0.0001

as I believed these parameters would make the network the most capable of learning complex patterns and information. After 500 epochs of training, this network had a RMSE of 0.890 by the last epoch and a reconstruction error of 0.790, which is still a relatively poor performance. However, this network was also able to identify a difference between the two simulation types in its latent space (Figure 27). Unlike the previous two latent spaces, this one has much less spread out data; the latent space appears to almost be the skeleton structure of a cube. This does make me slightly skeptical of this space in general (the odds of the latent space coincidentally forming this shape seems unlikely). However, it's impossible to know exactly what patterns the autoencoder is learning, so, assuming nothing was flawed in this methodology, this latent space indicates that, in general, the mutated simulation data differs from the wild type simulation data along the second latent space dimension. It also divides that data into possible clusters along the third dimension (or 'z axis') as the data for both simulation types appear to be split into two main groups along this axis. Lastly, the crystal structure again falls into the mutated simulation data area in this latent space.

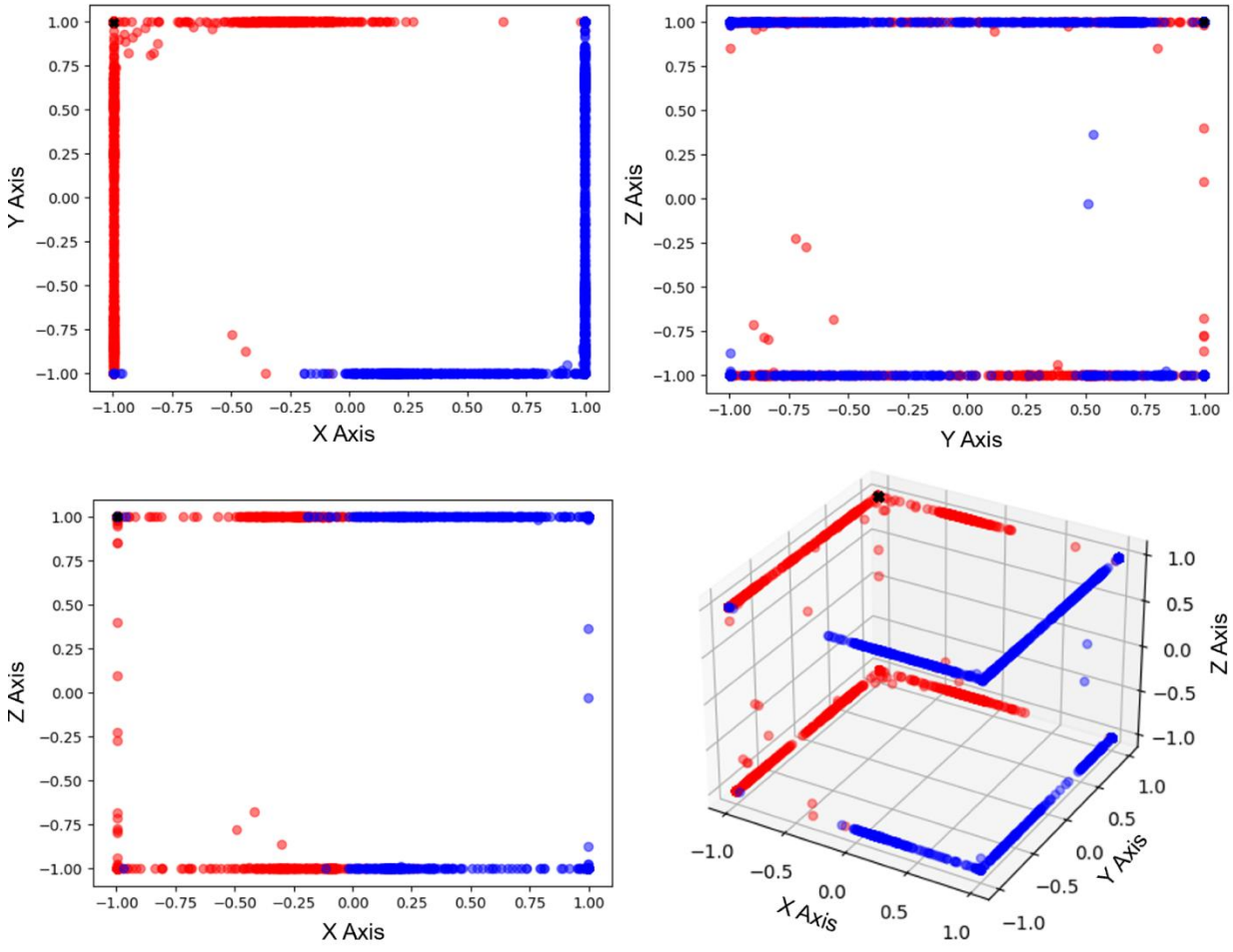


Figure 27: The latent space from the deep Tanh network with the encoded wild type data (blue), the encoded mutated data (red), and the encoded starting

structure (black X). The amount of simulation data plotted was reduced by taking every 25th data point to improve clarity.

Though I currently do not know how to analyze the latent spaces structurally, the fact that all networks were able to pick up on noticeable differences does indicate to me that there are structural differences between the two types of simulations. The fact that these differences were this obvious, even with relatively poor RMSEs for the networks is also impressive. (I also do not believe it is possible that these differences are a product of a poorly performing network, as I would image the "sorting" that is being done could not happen randomly.) In the future, it might be interesting to try to use the decoder to generate example structures to try to clarify what these dimensions are picking up on, though I image these differences might be subtle and hard to observe. It would also be interesting to see how these latent spaces, especially the ReLU one with higher overlap, change with a full-length simulation and removed equilibration data. There were also many other things I wanted to try in order to increase the performance of the networks, which I did not get to, including trying a sparse autoencoder, a sigmoid output layer, hidden layers of all the same size, and a leaky ReLU activation function along with more epochs.

9. The Effects of the Mutation

Based on the dimensionality reduction techniques explored throughout the previous chapters, mutating the Vif K50 residue appeared to have some impacts on the complex throughout simulation time. Unfortunately, our main technique for isolating and comparing the primary global motions of the complexes throughout simulation time (PTRAJ PCA) appears to have some potential faults, so I am unable to pinpoint exactly what differences there are between the wild type simulations and the mutated simulations. However, even though I might not be able to quantify or describe what these differences may be, I believe that the work throughout previous chapters shows that these differences are present, and hopefully in the future they can be given more concrete meaning.

The PCA analysis from chapter six showcases different clusters that the simulation data falls into along the top four principal components. The presence of clusters in PCA means that there are certain conformations that the simulations sample and potentially move between. The differences between the clusters for the wild type data and the mutated simulation data imply that these different complexes are sampling different conformations, even if subtly, and future exploration might be able to inform us of what these differences entail or mean. The KPCA results from chapter seven also highlight how the wild type simulation data and the mutated simulation data might have some non-linear differences. Based on the RBF KPCA plots (Figure 21), I would guess that these differences might entail the mutated simulation data having greater variance in the correlated motions compared to the wild type data, though this is a tentative hypothesis that would need to be explored further. If the mutated simulation data were more flexible and had more global motions overall than the wild type data, I would also expect this to be observable from regular PCA plots, so the KPCA differences cannot be fully understood at this time. Similarly, the autoencoder latent spaces from chapter eight either effectively separates the two simulations (Figure 25, 27) or depicts the mutated simulation data as having much more variance along all three latent space dimensions (Figure 26). The former implies that there is some difference between the two throughout the majority of the simulation time and the latter might indicate that the mutated simulation data experiences more variance than the wild type data. This could reinforce the idea that the mutated simulation data has more variance in its correlated motions over simulation time, though it is not possible to know if this interpretation of the figures is correct at this point.

Alongside the dimensionality reduction analysis techniques explored, I also made contact maps to visualize the contacts between the Vif and A3F residues (Figure 28). These maps were made using the last 300 ns of 600 ns simulation data; this data was not used for the dimensionality reduction techniques as the saved simulation data files were faulty. (This just affected EloB and Eloc, and since Vif and A3F were not affected by this, the contact maps reflect accurate simulations.) The Contact Map Explorer library was used to create these maps;

two residues are considered “in contact” by default if they are within 0.45nm of each other and every tenth simulation frame was read in[38]. When looking at the difference in contacts between the two proteins from the wild type and mutated simulation, it is clear that, at the site of the salt bridge (Figure 3), there are less contacts throughout the simulation after the K50E mutation was made. This indicates that the mutation from lysine to glutamate disrupted the salt bridge, consequently decreasing contacts between Vif and A3F at this site. (Vif residue 50 is 240 in the contact map numbering scheme.). Additionally, there are other decreases in Vif-A3F contacts near the site of the mutation, mainly concentrated around A3F residues one and 60. It is likely that the mutation also inhibited binding at these sites, though future study could look more into the interactions at these sites.

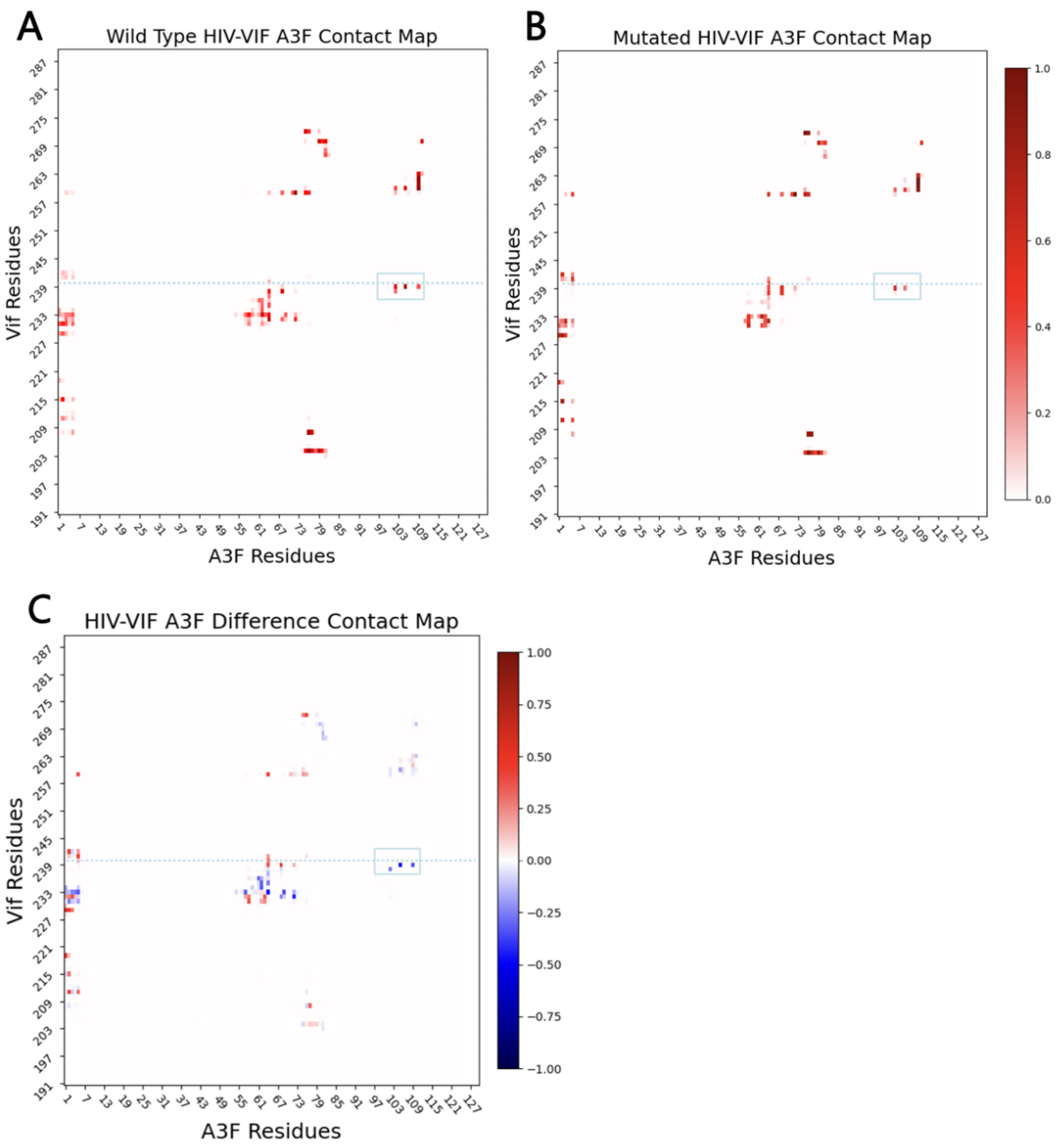


Figure 28. A) Interactions between Vif and A3F residues throughout simulation time for the wild type Vif-A3F complex and B) the mutated complex. C) Contacts that are present in the wild type but not the mutated complex are shown in blue. The dotted blue line indicates the location of the mutation, and the box indicates the site of the salt bridge. The color scale from 0.0-1.0 represents the percentage of simulation time any two residues are in contact.

Overall, the Vif K50E mutation appears to have impacted the binding site between Vif and A3F, especially at the location of the salt bridge (Figure 3). The mutation also seems to have impacted the VCBC-A3F complex's structure throughout simulation time, as differences between the wild type simulation data and the mutated simulation data were highlighted by the dimensionality reduction techniques. What exactly these differences are and whether or not they were significant enough to effectively prevent the binding of A3F and Vif requires future study. To better understand the differences structurally (especially with regards to the Vif-A3F binding site), next steps might be running clustering algorithms on the PCA results or the autoencoder latent spaces. Then, average structures could be visualized and compared. Additionally, contact maps could be made for each cluster to compare how many Vif residues are in contact with A3F. If some clusters have fewer contacts at the binding site, these could be reflective of inhibited binding. Furthermore, if the mutated simulations sample this cluster more than the wild type simulations, that could indicate that the mutation was able to successfully inhibit binding.

10. Conclusions

The long-term goal of this research is to better understand the VCBC-A3F complex; more specifically, with increased understanding of the Vif-A3F binding site it is possible to aid in the long-term development of therapeutics that could potentially inhibit this binding process. If successfully inhibited, Vif would not be able to bind with A3F after hijacking the VCBC complex, and consequently human A3F proteins would not be marked with ubiquitin and subsequently depredated. This would allow the A3F proteins to lead a sufficient immune response against HIV, rendering it harmless to the human body. Identifying key residues in Vif or the VCBC-A3F complex that facilitate the binding process could help with the future development of an inhibitor.

We were able to use free energy analysis to determine one such potential residue, and after computationally altering Vif residue K50, I was able to use molecular dynamics simulations to model how the mutation of this specific residue impacts the VCBC-A3F complex and the Vif-A3F binding site. Due to the high dimensional nature of the simulation data, some form of dimensionality reduction is required to isolate important features in the data to effectively analyze it. Through the use of principal component analysis, kernel PCA, and a trained autoencoder, I was able to reduce the dimensionality of 300 ns of wild type simulation data and 300 ns of the mutated simulation data. The differences between these simulation types in these reduced dimensionality spaces indicate that there are some inherent differences between the complex's structure throughout the wild type simulations and throughout the mutated simulations. Unfortunately, at this time, I am unable to quantify exactly what these differences entail structurally, but the nature of their existence indicates the mutation made did have an impact on the simulations. Future analysis should allow for insight into structural differences in the protein conformational ensembles.

In order to determine how well a specific dimensionality reduction technique is suited for MD simulation data, three criteria can be considered: 1) how fast it is, 2) how informative the low dimensional representation is and how well it separates the data into distinct states, and 3) how the high dimensional and low dimensional representations are linked[13]. Out of all the dimensionality reduction techniques explored in this paper, PCA was definitely the fastest to run, regardless of methodology used to run it. The autoencoders, when run on a GPU, took between 4-10 hours to train to completion depending on the duration used to train and the structure (though typically running for longer tends to yield better performance). However, once a network is trained, applying it to new data is easy and time efficient. The KPC ran the slowest; originally, when all data was included, this method was running for five days with no signs of stopping. The data was then significantly cut down, resulting in scripts that ran on the scale of minutes.

It is currently difficult to compare how informative the low dimensional representations of the data are between methods since there has not been enough time to explore what the representations mean. However, the PCA results appear to be best at separating the data into visibly distinct clusters. The KPCA might have done this to a lesser degree (in Figures 21a and 22 the wild type data might be split into two clusters), and the autoencoder seemed to separate the two different simulations into separate clusters. It might also be arguable that the deep Tanh autoencoder separated the data into four clusters with the wild type data and the mutated simulation data each splitting into two groups along the z-axis. Regardless, at this point it is not possible to confidently draw conclusions about how informative these representations are. One of the main appeals of using PTRAJ to run PCA is that it becomes obvious what the low dimensional data means, and consequently the representation becomes more informative about the original MD simulation data. However, now that I might have identified a potential flaw in the way we have been running this analysis, it might be worth looking more into either how to fix it, or make the other methods as informative. Similarly, I have not had any time to look into the link between the input data and the low dimensional representation. For some of the autoencoders, there appears to be a link between where a data point falls on the latent space and its original 'mutation status,' but what this entails and what other qualities to determine the latent space are unknown. Since so much is still unknown, this research is full of potential for future study, but hopefully this provides an introduction to these methods and their implementation for future researchers. Based on this work, I believe PCA and the use of autoencoders provide unique benefits for visualizing high dimensional data and KPCA is less informative and hard to interpret for MD simulation data. Additionally, in the future it might be interesting to look into Time-lagged independent component analysis (TICA), which is another dimensionality reduction technique that takes time-dependent information into account (as opposed to using each time frame simply as a data point).

Though the techniques explored in this paper are not sufficient to determine whether the mutation made to Vif residue K50 would successfully inhibit binding between Vif and A3F, it does show promise. Based on the differences between the simulation types observable in the reduced dimensionality visualizations and the decreased contacts between Vif and A3F, it appears to be worth looking into this residue further. If future research were able to further explore and quantify these differences, this residue could be a promising one to target in the development of therapeutics to prevent the replication of HIV in the body.

Acknowledgments

I'd like to thank Dr. Ball for providing me with the invaluable opportunity to dive into this research project four years ago and helping facilitate my learning and growth into an independent researcher. Thank you, Professor Tom O'Connell, for helping me this past year dive into the world of machine learning and dimensionality reduction and for constantly questioning me, supporting me, and helping me overcome the countless hurdles we encountered along the way. Juan Alcantara for building the starting structure used for simulations and Elizabeth Miller for running the free energy analysis used to identify the residue to mutate. Thank you, Matt Krzystyniak, for first mentoring me on simulation fundamentals when I joined this lab four years ago. Thank you, Beilynn Geiss, for continuing my research—it has been an honor and privilege to mentor you. Thank you, Michaela Cohen, for always being there to help me debug. And thank you to my supportive friends and family who got me through the numerous twelve-hour workdays and disheartening setbacks. I'd also like to thank the Student Opportunity Funds of Skidmore College for providing me with the funding to present this research at the Biophysical Society 2025 Annual Meeting and the 2025 New York Celebration of Women in Computing.

Glossary

(Protein) Backbone - a continuous chain of atoms that runs throughout the length of a protein consisting of a repeated sequence of nitrogen, alpha-carbon, and carbon atoms [39].

(Protein) Complex - a collection of proteins that interact with each other and have essential roles in cellular functions, regulatory processes, and signaling cascades [40].

(Protein) Conformation - the spatial arrangement of the atoms which make up a protein and consequently determine its overall shape; due to the flexibility of proteins (especially IDPs), conformations change rapids and are often linked to biological functions [41].

(Protein) Degradation – protein recycling in the cell.

Ensemble – a collection of states of a system typically obeying a Boltzmann distribution at equilibrium.

(Gibbs) Free Energy - a combination of enthalpy and entropy which can predict if a reaction is spontaneous or nonspontaneous given constant temperature and constant pressure; effectively, a decrease in free energy equates to a reaction that will occur spontaneously as the overall effect is stabilizing and visa versa [42].

Oligomer – multiple copies of the same protein that aggregate to form a complex.

Salt Bridge - bond between oppositely charged residues that are close enough to each other to experience electrostatic attraction; they contribute to protein structure and affect the interaction between proteins with other biomolecules [43].

Ubiquitinate - to covalently attach a ubiquitin protein to another protein; typically, this marks the protein for degradation.

References

1. Shrestha UR, Smith JC, Petridis L. Full structural ensembles of intrinsically disordered proteins from unbiased molecular dynamics simulations. Commun Biol 2021 February 23;4(1):1–8.
2. Mishra PM, Verma NC, Rao C, Uversky VN, Nandi CK. Intrinsically disordered proteins of viruses: Involvement in the mechanism of cell regulation and pathogenesis. Prog Mol Biol Transl Sci 2020 March 2;174:1–78.
3. Fajardo-Ortiz D, Lopez-Cervantes M, Duran L, Dumontier M, Lara M, Ochoa H, Castano VM. The emergence and evolution of the research fronts in HIV/AIDS research. PLoS One 2017 May 25;12(5):e0178293.
4. Vidya Vijayan KK, Karthigeyan KP, Tripathi SP, Hanna LE. Pathophysiology of CD4+ T-cell depletion in HIV-1 and HIV-2 infections. Front Immunol 2017 -05-23;8:580.
5. Sankaranantham M. HIV – is a cure possible? Indian J Sex Transm Dis AIDS 2019;40(1):1–5.
6. Kim DY, Kwon E, Hartley PD, Crosby DC, Mann S, Krogan NJ, Gross JD. CBFβ stabilizes HIV vif to counteract APOBEC3 at the expense of RUNX1 target gene expression. Molecular Cell 2013 February 21;49(4):632–44.
7. Ball KA, Chan LM, Stanley DJ, Tierney E, Thapa S, Ta HM, Burton L, Binning JM, Jacobson MP, Gross JD. Conformational dynamics of the HIV-vif protein complex. Biophys J 2019 April 23;116(8):1432–45.
8. Sadeghpour S, Khodaee S, Rahnama M, Rahimi H, Ebrahimi D. Human APOBEC3 variations and viral infection. Viruses 2021 July 14;13(7):1366.
9. Richards C, Albin JS, Demir Ö, Shaban NM, Luengas EM, Land AM, Anderson BD, Holten JR, Anderson JS, Harki DA, et al. The binding interface between human APOBEC3F and HIV-1 vif elucidated by genetic and computational approaches. Cell Reports 2015;13(9):1781–8.
10. Evans SL, Schön A, Gao Q, Han X, Zhou X, Freire E, Yu X. HIV-1 vif N-terminal motif is required for recruitment of Cul5 to suppress APOBEC3. Retrovirology 2014 January 14;11(1):4.
11. Hollingsworth SA, Dror RO. Molecular dynamics simulation for all. Neuron 2018 September 19;99(6):1129–43.
12. Tribello GA, Gasparotto P. Using dimensionality reduction to analyze protein trajectories. Front Mol Biosci 2019 June 19;6.
13. Lemke T, Peter C. EncoderMap: Dimensionality reduction and generation of molecule conformations. J Chem Theory Comput 2019 January 11;15(2):1209–15.
14. Nussinov R, Liu Y, Zhang W, Jang H. Protein conformational ensembles in function: Roles and mechanisms. RSC Chem Biol 2023 September 5;4(11):850–64.

15. Hu Y, Desimmie BA, Nguyen HC, Ziegler SJ, Cheng TC, Chen J, Wang J, Wang H, Zhang K, Pathak VK, et al. Structural basis of antagonism of human APOBEC3F by HIV-1 vif. Nat Struct Mol Biol 2019 -12-2;26(12):1176–83.
16. Guo Y, Dong L, Qiu X, Wang Y, Zhang B, Liu H, Yu Y, Zang Y, Yang M, Huang Z. Structural basis for hijacking CBF-β and CUL5 E3 ligase complex by HIV-1 vif. Nature 2014 January 8;505(7482):229–33.
17. Humphrey W, Dalke A, Schulten K. VMD: Visual molecular dynamics. J Mol Graph 1996;14(1):33–8.
18. Webb, , Benjamin, Sali, Andrej . Comparative protein structure modeling using MODELLER. Curr. Protoc. Bioinform. 2016 June 20 (54: 5.6.1-5.6.37).
19. Case DA, Aktulga HM, Belfon K, Cerutti DS, Cisneros GA, Cruzeiro VWD, Forouzesh N, Giese TJ, Götz AW, Gohlke H, et al. AmberTools. J Chem Inf Model 2023 October 23;63(20):6183–91.
20. D.A. Case, H.M. Aktulga, K. Belfon, I.Y. Ben-Shalom, J.T. Berryman, S.R. Brozell, D.S. Cerutti, T.E. Cheatham, III, G.A. Cisneros, V.W.D. Cruzeiro, T.A. Darden, N. Forouzesh, M. Ghazimirsaeed, G. Giambaşu, T. Giese, M.K. Gilson, H. Gohlke, A.W. Goetz, J. Harris, Z. Huang, S. Izadi, S.A. Izmailov, K. Kasavajhala, M.C. Kaymak, A. Kovalenko, T. Kurtzman, T.S. Lee, P. Li, Z. Li, C. Lin, J. Liu, T. Luchko, R. Luo, M. Machado, M. Manathunga, K.M. Merz, Y. Miao, O. Mikhailovskii, G. Monard, H. Nguyen, K.A. O'Hearn, A. Onufriev, F. Pan, S. Pantano, A. Rahnamoun, D.R. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, A. Shajan, J. Shen, C.L. Simmerling, N.R. Skrynnikov, J. Smith, J. Swails, R.C. Walker, J. Wang, J. Wang, X. Wu, Y. Wu, Y. Xiong, Y. Xue, D.M. York, C. Zhao, Q. Zhu, and P.A. Kollman. Amber 2024 [computer program]. University of California, San Francisco: 2024.
21. Pang YP, Xu K, Yazal JE, Prendergas FG. Successful molecular dynamics simulation of the zinc-bound farnesyltransferase using the cationic dummy atom approach. Protein Sci 2000 -10;9(10):1857–65.
22. 1. Prepare Input File [Internet]. Available from: https://ambermd.org/tutorials/advanced/tutorial8/loop3.php.
23. Lysine [Internet]National Library of Medicine. Available from: https://pubchem.ncbi.nlm.nih.gov/compound/5962.
24. Glutamate [Internet]National Library of Medicine. Available from: https://pubchem.ncbi.nlm.nih.gov/compound/4525487.
25. Deisenroth MP, Faisal AA, Ong CS. Mathematics for machine learning. Cambridge University Press; 2020.
26. Palma J, Pierdominici-Scottile G. On the uses of PCA to characterise molecular dynamics simulations of biological macromolecules: Basics and tips for an effective use. Chemistry Europe 2022 Oct 26.
27. Tribello GA, Gasparotto P. Using dimensionality reduction to analyze protein trajectories. Front Mol Biosci 2019 June 18;6.

28. Roe DR, Cheatham TEI. PTRAJ and CPPTRAJ: Software for processing and analysis of molecular dynamics trajectory data. J Chem Theory Comput 2013 June 10;9(7):3084–95.
29. RMSD Analysis in CPPTRAJ [Internet]; c2014. Available from: https://ambermd.org/tutorials/analysis/tutorial1/index.php.
30. Lindholm A, Wahlström N, Lindsten Fredrik, Schön TB. Machine learning - A first course for engineers and scientists. Cambridge University Press; 2022.
31. Briscik M, Dillies M, Déjean S. Improvement of variables interpretability in kernel PCA. BMC Bioinformatics 2023 July 12;24(1):282.
32. KernelPCA [Internet]. Available from: https://scikit-learn/stable/modules/generated/sklearn.decomposition.KernelPCA.html.
33. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.
34. Vert J, Tsuda K, Schölkopf B. A primer on kernel methods. In: Kernel methods in computational biology. The MIT Press; 2004.
35. 6.8. Pairwise metrics, Affinities and Kernels [Internet]. Available from: https://scikit-learn/stable/modules/metrics.html.
36. Mansoor S, Baek M, Park H, Lee GR, Baker D. Protein ensemble generation through variational autoencoder latent space sampling. J Chem Theory Comput 2024 April 9;20(7):2689–95.
37. Degiacomi MT. Coupling molecular dynamics and deep learning to mine protein conformational space. Structure 2019 -06-04;27(6):1034,1040.e3.
38. Contact Maps [Internet]. Available from: https://contact-map.readthedocs.io/en/docs/examples/nb/contact_map.html#.
39. Si D, Moritz SA, Pfab J, Hou J, Cao R, Wang L, Wu T, Cheng J. Deep learning to predict protein backbone structure from high-resolution cryo-EM density maps. Sci Rep 2020 March 9;10(1):4282.
40. He Z. 7 - protein complex identification from AP-MS data. In: Zengyou He, editor. Data mining for bioinformatics applications. Woodhead Publishing; 2015.
41. Gupta A, Singh A, Ahmad N, Singh TP, Sharma S, Sharma P. Chapter 12 - experimental techniques to study protein dynamics and conformations. In: Timir Tripathi, Vikash Kumar Dubey, editors. Advances in protein molecular and structural biology methods. Academic Press; 2022. ID: 781595.
42. Gibbs (Free) Energy [Internet]; c2013. Available from: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Thermodynamics/Energies_and_Potentials/Free_Energy/Gibbs_(Free)_Energy.
43. Bosshard HR, Marti DN, Jelesarov I. Protein stabilization by salt bridges: Concepts, experimental approaches and clarification of some misunderstandings. J Mol Recognit 2004;17(1):1–16.