**DECLARATION BY THE STUDENT**

I hereby declare that the thesis “*Developing a Reinforcement Learning Model for Somatic Hypermutations using PD1 and Pembrolizumab*” is a bona fide and genuine research work carried out by me, between 23rd December, 2024 and 4th June, 2024 at IBAB, Bengaluru, under the guidance of *Dr. Nithya Ramakrishnan*, Assistant Professor, Information Theory, Algorithms and Machine Learning in Biology and *Prof. Subha Srinivasan*, IBAB Chair, Genomics.

Date:

Place: Anmol Singh

**CERTIFICATE BY THE SUPERVISOR**

This is to certify that the thesis “*Developing a Reinforcement Learning Model for Somatic Hypermutations using PD1 and Pembrolizumab*” represents research work done by *Anmol Singh* in partial fulfillment of the requirements for M.Sc. in Biotechnology and Bioinformatics at IBAB, Bengaluru, under our guidance.

Date:

Place: Dr. Nithya Ramakrishnan

Prof. Subhashini Srinivasan

**ACKNOWLEDGMENTS**

I express my deepest gratitude and special thanks to my guides Dr. Nithya Ramakrishnan and Prof. Subhashini Srinivasan who took time to hear me out, guide and keep me on the correct path and allowed me to bring the project to a good conclusion. Dr. Nithya Ramakrishnan helped me regarding the reinforcement learning, whereas Prof. Subhashini Srinivasan provided guidance related to structural biology and protein-protein interactions, without it the project would not have a base to build upon.

I express my thanks to the lab members of the research groups of my guides, especially Mr Balakrishna, Miss Namita and Miss Apoorva who helped me during the project and gave important and necessary advice to overcome the problems I faced in the project and life during this time.

I thank all the professors who helped me acquire the required knowledge and understanding of biology, mathematics and computer science which helped me to confidently work on this project. I especially thank Prof. R Srivatsan, our mathematics, statistics and R programming professor, for his guidance and teachings during my masters.

I would also like to thank my mother Mrs. Rekha Singh for always listening to my rambles about this project and by explaining my work to her, I have acquired a better understanding of reinforcement learning and structural biology. Needless to say, I would not have the courage to pursue my masters if not without her supporting and believing in me. This thesis is dedicated to her.

This work was partially supported by the Department of Electronics, IT, BT, and S&T of the Government of Karnataka and I am grateful for their support.

**ABSTRACT**

The process of engineering antibodies with high affinity towards an antigen is very expensive and time taking in a wet lab. Bioinformatics tools can be used to reduce the time and money required. Reinforcement learning is similar to natural selection, as in by trial-and-error method, the better actions(mutations) remain in the population and those harmful are removed over time. We present a somatic hypermutations(SHM) reinforcement learning model which can learn to preferentially mutate amino acids on antibodies and lead to affinity maturation, predicting a higher binding affinity antibody than the starting antibody-antigen complex. We have used the Pembrolizumab-PD1 (5b8c) complex to create the model, as Pembrolizumab is widely used in immunotherapy. We were able to use Q-Learning in reinforcement learning to model SHM on a reduced state space to and provide better binding affinity antibodies. We validated the structure of the antibodies predicted by the SHM model by using AlphaFold2 and inter residue distance plots to check for proper folding of chains and protein-protein interactions. This study provides a proof of concept that reinforcement learning can be used for modeling the biological process of SHM and can be later expanded into creating novel antibodies.

**Keywords or phrases (Max. 10):** Antibody prediction, Q-learning, Deep Q-Learning, Deep Reinforcement Learning, AlphaFold2, protein-protein interaction, immunotherapy

**TABLE OF CONTENTS**

| Content | Page Number |
| --- | --- |
| 1. Introduction | 1-12 |
| * 1. Somatic Hypermutations | 1-2 |
| * 1. Immunotherapy and Pembrolizumab | 2-3 |
| * 1. Binding Affinity | 4-5 |
| * 1. Reinforcement Learning | 5-10 |
| * 1. Deep Reinforcement Learning | 10-12 |
| * 1. Objectives | 12 |
| 1. Materials and Methods | 12-25 |
| * 1. Pembrolizumab | 12-14 |
| * 1. Binding Affinity Tools | 14-15 |
| * 1. Basic Q-Learning | 15-21 |
| * 1. Deep Q-Learning | 21-23 |
| * 1. Structure Validation | 23-25 |
| 1. Results and Discussion | 25-47 |
| * 1. State Aggregation | 25-27 |
| * 1. Mutating residues of Light Chain | 27-31 |
| * 1. Mutating residues of Heavy Chain | 31-38 |
| * 1. Deep Q Learning | 38-40 |
| * 1. Validation of Higher Affinity States | 40-47 |
| 1. Conclusions | 47 |
| 1. References | 48-49 |

**List of figures**

| No table of figures entries found. |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**List of tables**

| No table of figures entries found. |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

1. **INTRODUCTION**

It is important to introduce some of the basic terms and concepts, these mainly answer specific questions:

* What are somatic hypermutations? Why are they important for our immune system?
* What is immunotherapy and how is this study related to it? Where is pembrolizumab used in immunotherapy?
* What is reinforcement learning? Why is it a good strategy for simulation SHM?
  1. **Somatic Hypermutations**

The B cells have the ability to express cell-surface receptors called immunoglobulins (Ig). These Ig consists of two heavy chains and two light chains. These polypeptides are encoded in three Ig loci, the heavy chain (IgH), the κ-light chain (Igκ), and the λ-light chain (Igλ). These loci consist of variable and constant(C) regions. The variable region of the heavy chain is composed of variable(V), diversity(D), and joining(J) genes. On the other hand, the light chain only has V and J genes (Odegard and Schatz, 2006).

Our Immune system has a wonderful system to produce Ig of high specificity towards any foreign substance, also called as antigen. This ability is the result of somatic recombination of a small set of gene segments; this process is called V(D)J recombination. This process alone is able to produce around 107 different antibody specificities. But the antibodies created by V(D)J recombination only bind to the antigens by modest affinity, there is a need to fine tune the resultant Ig to make it bind to the antigens with high affinity and specificity (Papavasiliou and Schatz, 2002).

The diversification of Ig is caused due to two distinct diversification processes, class switch recombination (CSR) where the C region of IgH changes due to recombination of switch (S) regions, and the somatic hypermutations (SHM) (Odegard and Schatz, 2006).

The SHM introduces point mutations on the variable regions, the antibodies with higher affinity for the antigen will proliferate and survive. With successive cycles of mutations and proliferation of selected B cells, this results in high affinity antibodies. This process is called affinity maturation (Papavasiliou and Schatz, 2002). The mutations in SHM are mainly point mutations, but insertions and deletions are also observed sometimes (Odegard and Schatz, 2006).

For my model, I have only taken point mutations into consideration, on the peptide level rather than gene level.

* + 1. **Assumptions related to SHM model**

When considering we are working with only somatic hypermutations, let us make few assumptions for the model:

* The relative configuration of PD1-Pembrolizumab complex remains the same irrespective of the type of amino acids mutations at the 17 positions
* The 3D folds of light and heavy chains are not disrupted by the mutations in loops on pembrolizumab.
  1. **Immunotherapy and Pembrolizumab**

Immunotherapy is the ability to utilize a patient’s immune system to target cancer has resulted in many novel therapeutic ways. But even though these approaches are useful in many cases there are still many challenges in the clinical scenarios.

The tumor-host interactions are heterogenous and based on these interactions the immunotherapy responsiveness can differ. The tumor microenvironment (TME), can affect the immunotherapeutic response and the immune evasion.

There are multiple types of immunotherapies used currently. These are immune checkpoint inhibition (ICI), adoptive cellular therapy (CAR T-cell therapy) and cancer vaccination. Pembrolizumab comes under the ICI, thus let us understand it in more detail.

The T cells contain evolutionarily conserved regulatory markers that are like checkpoints to regulate activation of T cells. After the early activation, the T cells upregulate the inhibitory receptor cytotoxic T lymphocyte antigen 4 (CTLA4) and then after programmed cell death 1 (PD-1) which bind to ligands B7-1, B7-2 and PD-L1 or PD-L2. These ligands are presented by tumor cells, myeloid cells, regulatory T cells (Tregs), and antigen-presenting cells (APCs), which reduce the cytotoxic T-cell activation, resulting in immune suppression and tumor growth. But after treatment with ICI, inhibition is released and cancer cells are targeted and destroyed by the primed and activated cytotoxic T cells.

Pembrolizumab (pembro), a IgG4 anti PD1 checkpoint inhibitor antibody, was one of the first FDA-approved therapy for melanoma. Pembro was very successful in the melanoma patients and is still used in ICI therapy. I am using pembrolizumab as a model IgG antibody to perform SHM and possibly find a higher affinity antibody (Peterson et al., 2022).

* 1. **Binding Affinity**

For us to evaluate the antibody and antigen interactions quantitatively and qualitatively and to do that we need to understand the type of interactions which happen between Ab-Ag and their binding affinity.

By generalizing our Ab-Ag to protein-protein interaction (PPI) we can understand it better. Given the high-resolution structure, these interactions can be explained as a function of atomic distances and characteristics, but this does not describe the function. It is the affinity of these interactions under biological conditions, such as temperature, pH and protein and substrate concentration and how they bind actually describes if the atomic interactions are feasible.

The binding affinity is explained in terms of equilibrium dissociation constant (Kd), which can either be measured experimentally at equilibrium of reaction or be derived from the Gibbs free energy of dissociation ΔG and the reaction kinetics.

Experimentally there are several methods such as surface plasmon resonance (SPR), isothermal calorimetry (ITC), and titration by fluorescence which are applicable if Kd is in the micromolar to nanomolar range. Although these methods have limitations. Normally, two or more methods are used under similar conditions to determine the Kd.

But for a simulation such as ours, we neither have a dataset of earlier experimentally determined and neither is experimentally determining Kd feasible, computational methods of estimating the binding affinity from the given high-resolution structure is the only option.

These methods may use several parameters into account including the intermolecular non-covalent contacts (Kastritis P et al., 2011), and also the non-interacting surface (Kastritis P et al., 2014) and create linear or nonlinear models or machine learning models to estimate the binding affinity and compare them with the experimentally determined values to determine if their models work. I have described some of these methods under **§** 2.2.

* 1. **Reinforcement Learning**
     1. *Basic Overview*

Reinforcement learning in simpler terms is learning what to do in a situation to maximize a numerical reward. We normally explain the problem of reinforcement learning as Markov decision processes. The basic idea is that an agent which is interacting with an environment overtime is able to sense the state of the environment and take actions as to affect the state to achieve a goal over time **t**. Markov decision processes have multiple components: an Agent, the Environment, State of the environment, action taken by the agent and the reward. We use **S** for a set of states, **A** for a set of actions and **R** for a set of rewards (Sutton, R. S. and Barto A. G., 2018).

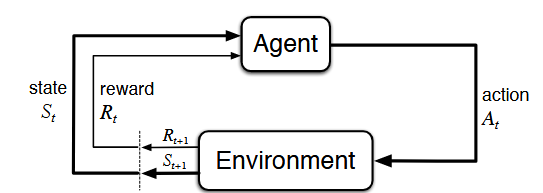


Figure: The interaction of agent and environment in a Markov decision process (Sutton, R. S. and Barto A. G., 2018)

It is important to note that an agent’s aim is to maximize the expected return of rewards. For our study the MDP consist of:

* Agent: PyMOL: creating mutations
* Action: Point mutations on Pembrolizumab
* Environment: Pembrolizumab-PD1 complex
* State: Pembrolizumab with mutated amino acids
  + - 1. *Episodic and Continuing tasks*

The tasks can also be of two types, episodic or continuing. When we can break the task into smaller episodes such as a game of Capture the Flag (CTF) where each episode ends when a team captures the flag of the opponent and the game refreshes from start. On the other hand, if the interactions are not broken naturally into episodes, we call it a continuing task like riding a bicycle (Sutton, R. S. and Barto A. G., 2018).

* + - 1. *Discounted Reward*

When we talk about return of reward, it normally means the discounted return of rewards (**Gt**). This way the agent will care about immediate rewards more than the future rewards (Sutton, R. S. and Barto A. G., 2018):.

This can be explained in terms of rewards at successive time steps:

Where .

* + - 1. *Policies and Value Functions*

In brief, policies address the problem of action selection in any given state. Whereas, value functions tell us how good a given action or state is for an agent.

Policies are simple functions which govern how the agent picks an action. The value functions are then defined according to the policies. There are different value functions (Sutton, R. S. and Barto A. G., 2018).

* + - * 1. *State-Value Function*

For a policy the state value function tells us how good the state is for an agent following the policy .

* + - * 1. *Action-Value Function*

Similar to the action-value function tells us for a given policy, how good it is for the agent to take any given action (**a**) in a given state(**s**) at time(**t**).

This is also called the Q-function and provides a Q-value and is an important component of the Q-Learning (Sutton, R. S. and Barto A. G., 2018).

* + - * 1. *Optimal Policy and Value Functions*

Briefly, a policy is considered to be optimal than another policy if,

Here stands for the expected return for starting in a state and then following the policy .

The optimal value functions are defined as:

Optimal State-Value function:

Optimal Action-Value function:

For all and **.**

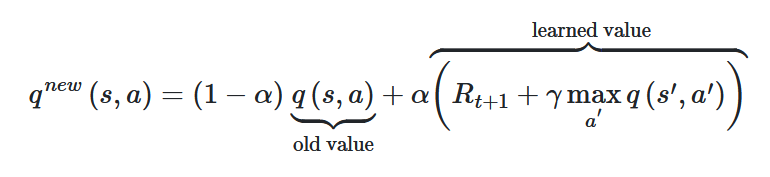
One fundamental property of is that it must follow Bellman optimality equation (Sutton, R. S. and Barto A. G., 2018) as follows:

* + 1. *Basic Q-Learning*

The basic Q learning algorithm to find the optimal policy is by finding the optimal Q-values for each state-action pair.

How do we achieve this? We utilize the Bellman optimality equation to find the optimal . We iterate through all the states and fill the Q-table which consists of the Q-value for the state-action pairs.

The Q-value is updated according to the learning rate(α), i.e. as the simulation goes on, the agent learns from the state-action pairs it has observed. This is calculated according to the given formula (Sutton, R. S. and Barto A. G., 2018):



Here we can observe that in this Q-value update equation, the learned value is the same as the Bellman optimality equation.

* + - 1. *Exploration methods*

One important part of any reinforcement algorithm is how to define the exploitation and exploitation are defined. Exploration is simply the agent trying to explore the environment. On the other hand, exploitation is taking action based on the policy, thus taking the best or one of the best actions in a given state. In terms of Q-learning, exploration is taking random selection of actions whereas exploitation is utilizing the learnt Q-values to determine the best action for a given state, i.e. exploitation of Q-table.

There are different exploration-exploitation methods. Let us discuss two of them in brief.

* + - * 1. *Epsilon greedy Strategy*

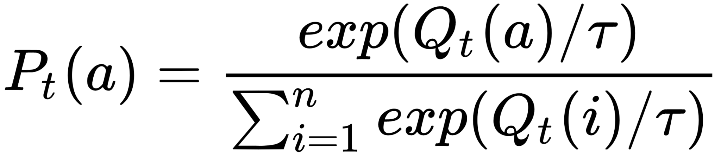
Here, we define an exploration rate which is initially set to 1(100%). Then in each episode, or in intervals of episodes we reduce this epsilon according to a decay rate. As the epsilon decreases, the agent slowly shifts from exploration of the environment to the exploitation of the Q-table (Sutton, R. S. and Barto A. G., 2018).

We do this by picking a random number in each instance and then comparing it with the exploration rate. If the random number is greater than epsilon then we go for exploitation, else we go for exploration.

The problem with the epsilon greedy method is that it cannot distinguish between two good actions/Q-values. It always picks the highest Q-value during exploitation. To overcome this there are other exploration methods, such as Soft-max strategy.

* + - * 1. *Soft-max (or Boltzmann) Strategy*

To overcome the shortcoming of the epsilon greedy method during exploitation, we can convert the Q-values into a Boltzmann distribution and then pick actions according to it. Here a hyperparameter temperature(τ) is introduced.



We start with a high temperature which corresponds to exploration and as we reduce the temperature the agent starts to exploit the Q-table (Sutton, R. S. and Barto A. G., 2018).

* + - 1. *Limitations of Q-learning*

Basic Q-learning is good when dealing with small state space of up to few hundred or thousand, but as we increase the number of possible states the agent can observe overtime, the algorithm starts to demand more exploration and thus takes much more time.

Most of the real-world problems have a very large state space, thus basic Q-learning is not effective for problems. To find solutions to such problems, we need to consider many other reinforcement learning approaches such as policy gradient methods, actor-critic, model-based methods and Deep Q Networks. There are multiple Q-Learning algorithms other than the basic one explained.

* 1. **Deep Reinforcement Learning (DRL)**

When we talk about deep learning we talk about utilizations of different Artificial Neural Networks (ANN). There are many types of deep reinforcement techniques but here I am only describing Deep Q-Learning (DQL) which was implemented in the work.

* + 1. *Artificial Neural Networks (ANN)*

ANN mainly consists of three types of neural layers, a layer of “input”, then comes the “hidden” layers and lastly the output layer. There are multiple types of ANN, one of the most commonly used ones are feedforward ANN (FFNN) and convolutional neural network (CNN). For our model we are working with FFNN, so let us explain it in brief. Although I must point out, for Deep Q-Learning the most commonly used ones are CNN, which was introduced to learn Atari games (Mnih V et al., 2013).

FFNN are the simplest types of ANN, the steps included in training a FFNN are:

1. Create a FFNN with randomized weights
2. Do a forward pass through the network and get the predicted value
3. Calculate the loss of predicted and expected value using functions such as mean square error (MSE)
4. Do backpropagation through the network to get the slope of loss vs weights; there can be a optimization algorithm which takes care of this such as Adam (Kingma D, 2017)
5. Do a gradient descent to find the nearest minima and adjust the weight accordingly
   * 1. *Deep Q Learning (DQL)*

For DQL we are replacing our policy with ANN, we call this a Deep Q-Network (DQN), which is now used to approximate optimal Q-values. This algorithm was explained by Mnih V et al., 2013, to create a DQL model to play Atari games, and has been used in many other complex games such as DOTA2 and GO. Although the original algorithm calls for a CNN as the games played needed to be analyzed, which cannot be done through a simple FFNN. CNN has been used for deep learning algorithms for data which includes images to be analyzed. The detailed algorithm of the DQL is explained later.

* 1. **Objectives**
     1. *Major Objective*

The main objective of this project is to create a reinforcement learning model for somatic hypermutations using pembrolizumab and PD1.

The reason we choose the reinforcement learning (RL) model is because unlike supervised and unsupervised learning, RL models don't require labeled data to train the model. RL is similar to natural learning of trial and error; we simultaneously generate the data and the model learns from it based on the mistakes depending on the action and reward received for the state.

* + 1. *Minor Objective*

We also aim to find states/antibodies which have better binding affinity than pembrolizumab, and validating the good states using AlphaFold2 and C-alpha – C-alpha distance plots of the PDB structures. This gives the study an applicative overview as we can provide theoretical states which may produce better antibodies in wet lab, here pembrolizumab for immunotherapy.

1. **MATERIALS AND METHODS (3000 words)**
   1. **Pembrolizumab**

The crystal structure of Pembrolizumab (Pembro) was acquired from RCSB Protein Data Bank with ID: 5B8C. The structure contains multiple complexes of Pembro-PD1, the complex consisting of chain A (light chain), B (heavy chain) and C (PD1) was taken. It was then relaxed using GROMACS (version 2020.1-Ubuntu-2020.1-1) with the following script:

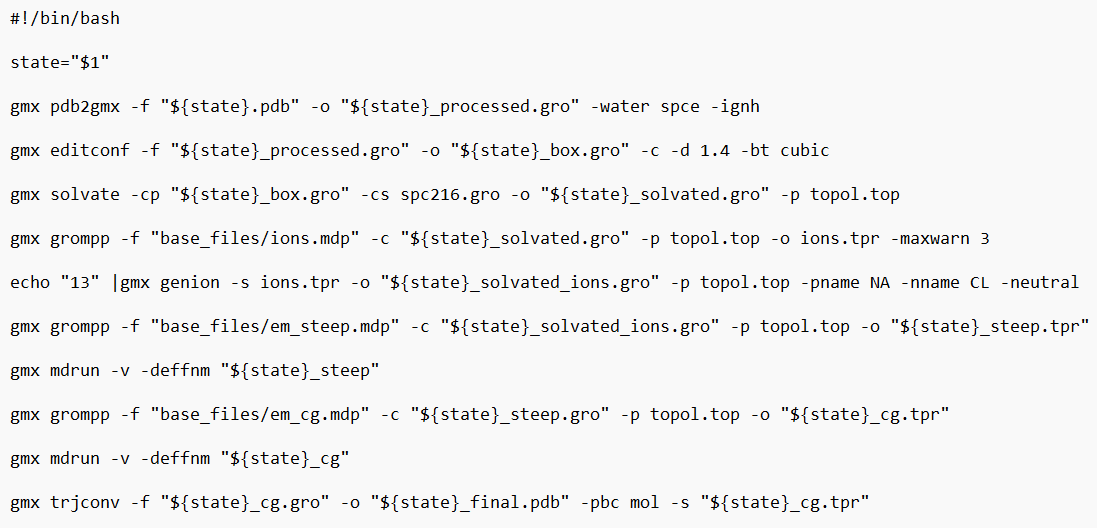


Figure: Bash script for running GROMACS for energy minimization

The “\*.mdp” files used had the following parameters:

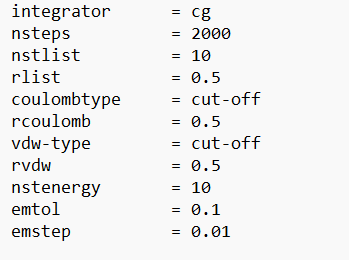
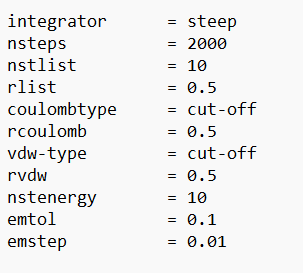
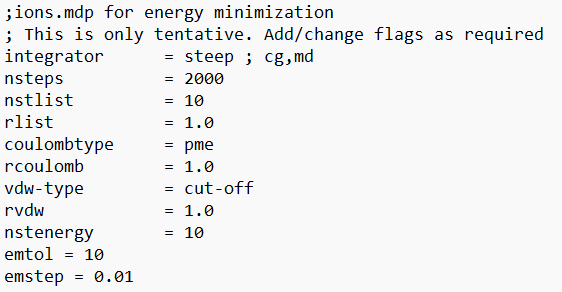


Figure: Parameters of ion.mdp(left), em\_steep.mdp(middle) and em\_cg.mdp(right)

We are working with and mutating 17 residues on Pembro (5b8c) which make contact with the PD1 (Horita S et al., 2016):

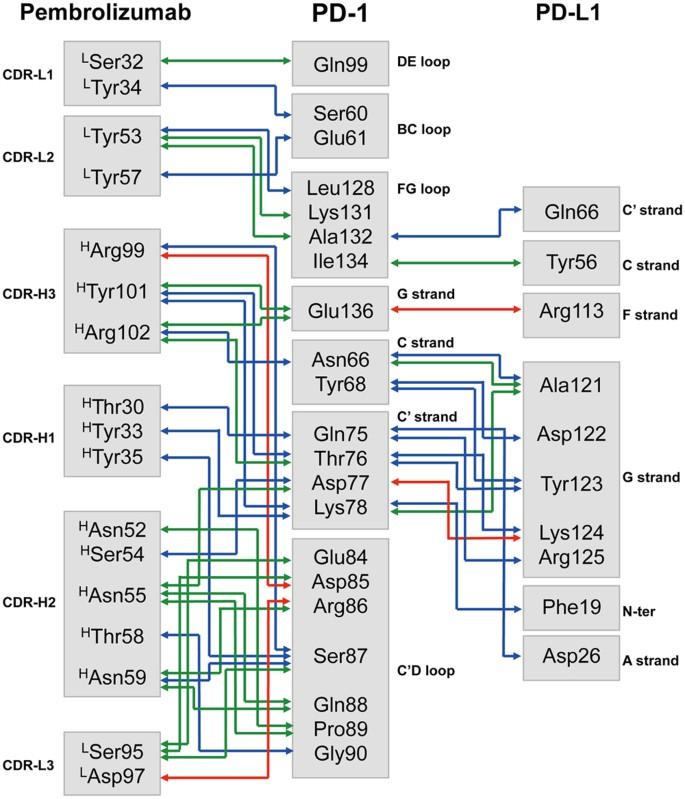


Figure: Interaction map of residues in Pembrolizumab-PD1 complex (5b8c)

* 1. **Binding Affinity Tools**

Binding affinity can be determined experimentally, but it isn’t feasible to do so, thus we use computational methods to estimate binding affinities. To choose which method is best for our use depends on: time taken to estimate binding affinity, accessibility of the method (if it is available via web-server or can be installed on local machines), accuracy and relevance of the method, and the metadata provided by the method. I compared multiple methods including PRODIGY (Xue L et al., 2016), LISA (Raucci R et al.,2018), CSM-AB (Myung Y et al., 2022), AREA-AFFINITY (Yang Y et al, 2023) and DG-Affinity (Yuan Y et al,2023) by literature review and comparing them based on the parameters mentioned before.

PRODIGY uses a linear model consisting of types of intermolecular non covalent contacts and also non interacting surfaces (NIS) to predict the binding affinity. LISA utilizes a non-linear model to estimate the protein-protein interactions. Both tools are considered good for protein-protein interactions (PPI) (Yang Y et al, 2023). They are locally installable and fast. But LISA has not been maintained and not dependable for the future. PRODIGY is maintained (<https://github.com/haddocking/prodigy>) and also provides metadata on the type of contacts the proteins are making.

CSM-AB, AREA-AFFINITY and DG-Affinity are web-server based tools and are specifically made for Antibody and antigen interaction. But AREA-AFFINITY and DG-Affinity do not provide an API for easy accessibility. CSM-AB is slow and the web-server goes off-service occasionally, which is problematic when running long simulations.

PRODIGY (version 2.1.3) was considered the best for our use case after the assessment of tools. The command used for prodigy:

$prodigy {pdb\_file} --selection A,B C --temperature 25

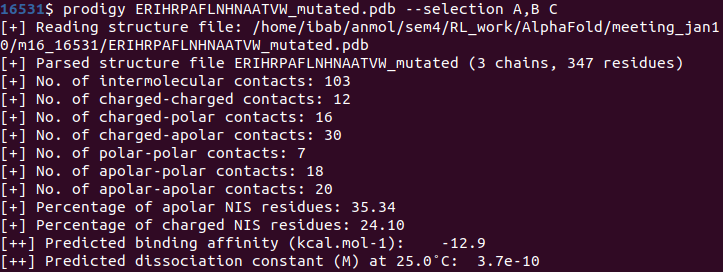


Figure: Example output from PRODIGY for a mutated state

The residues are categorized as follows in PRODIGY:

* Polar: C, H, N, Q, S, T, W
* Apolar: A, F, G, I, L, V, M, P, Y
* Charged: E, D, K, R
  1. **Basic Q-Learning**
     1. *Basic Workflow*

The basic pipeline for the Q-Learning is as follows:

1. Initiate the Brain class with the action list
   1. Action list consists of the all the possible actions
   2. Create an empty dictionary of Q-table with the action list
   3. Assign the hyperparameters such as learning rate, exploration rate, etc
2. Initiate the simulation with the starting state of Pembro
3. Predict action through Epsilon greedy or Soft-max strategy
4. Perform the mutation in PyMOL (version 2.6.0a0 Open-Source, 2024-01-19) and save the state in a local PDB bank for simulations
5. Use PRODIGY (version 2.1.3) to get the interactions and binding affinity of mutated state
6. Calculate the Score for the mutated state, and convert the score into discrete rewards. Score is generated by doing a dot product of input from PRODIGY and scoring vector.

* Scoring Vector: [ 0.1062428, -0.1065163, 0.396131, -0.3537632, 0.6397215, -0.1923142, -0.3012518, -0.1059661, -0.1]; in some simulations the -0.1 is changed to -0.5 or -2 (mentioned where it is done).
* Scoring vector was extracted from Xue L et al., 2016 and altered to add van-der Waals as negative reward, and charged-polar and apolar-apolar contacts were added according to their Pearson correlation with binding affinity.

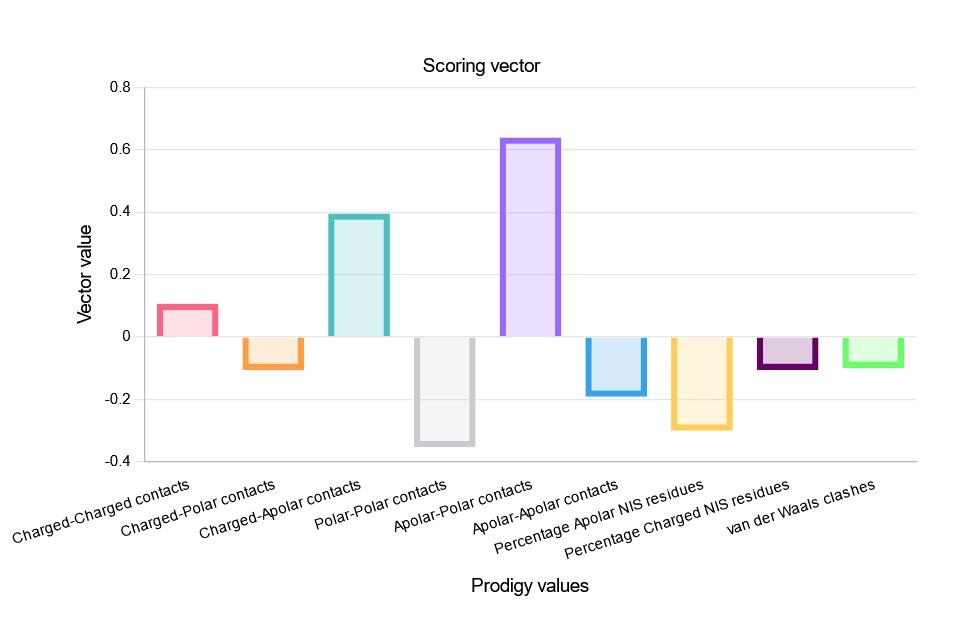


Figure: Scoring Vector

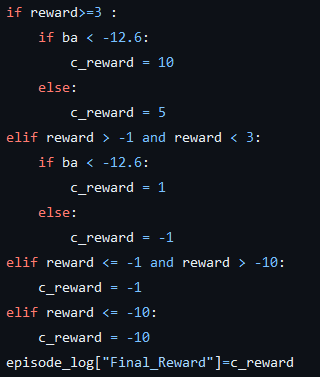


Figure: Converting score (“reward”) into reward (“c\_reward”)

1. Update the Q-value according to the Q-value update equation (**§**1.4.2).
2. Iterate through step 3 to 7 till simulation reaches the maximum episodes limit.

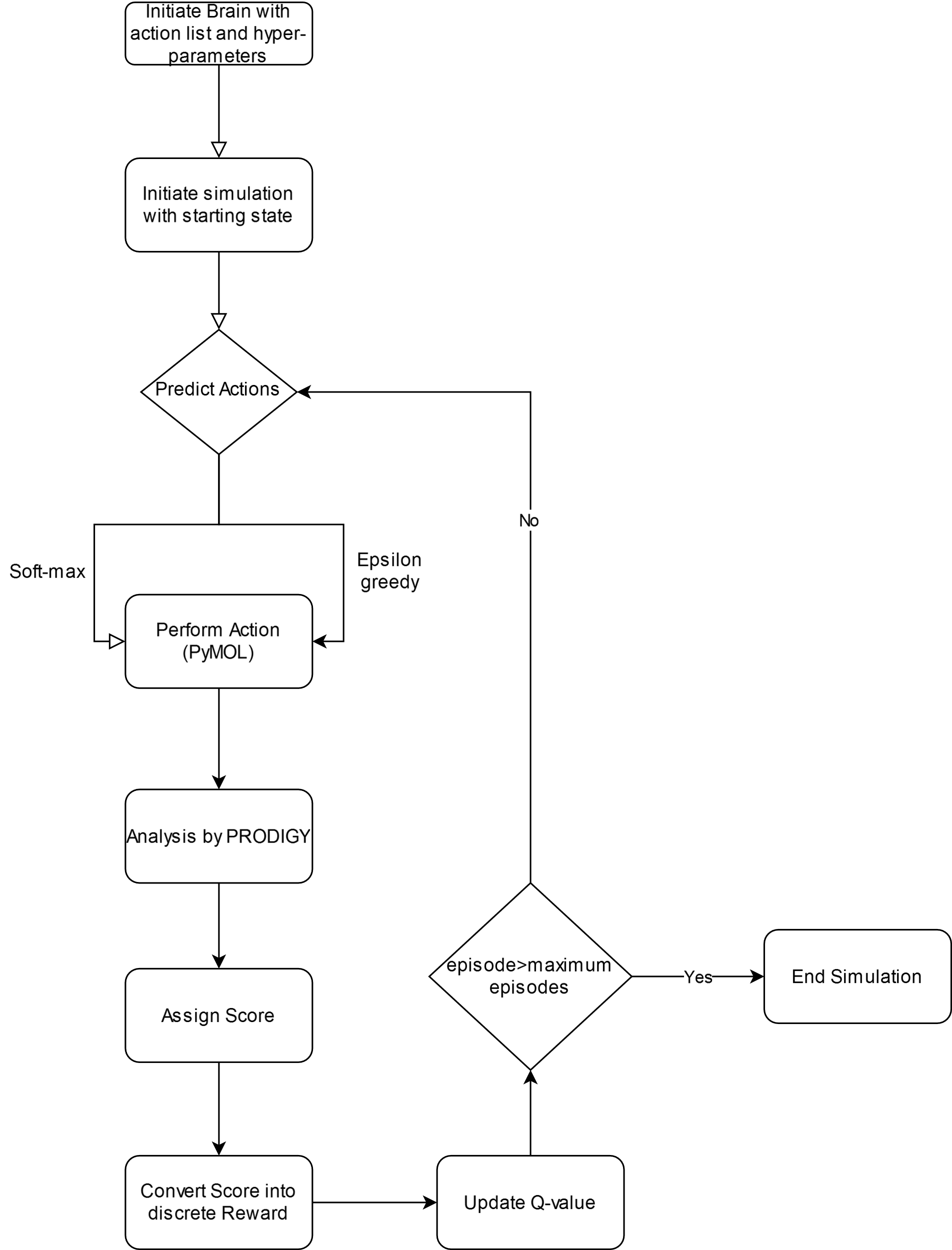


Figure: Flowchart of Basic Q-learning

* + 1. *State Aggregation*

A major problem with our problem statement is that for 17 residues to be mutated into 20 standard amino acids the state space is 1720, which is very huge to explore. As such basic Q-learning will not work. But if the states are aggregated based on common properties, we can reduce the state space to become more explorable.

To perform state aggregation, the following pipeline was used:

1. Perform simulations to collect data, I ran two simulations of different random seeds.
2. Perform PCA on the data collected using PCA package of sklearn v. 1.4.2
3. Perform k-means clustering on the PCA (using KMeans package of sklearn)
   1. Random seed for 600 centroids: 42
   2. Random Seed for 10000 centroids: 66

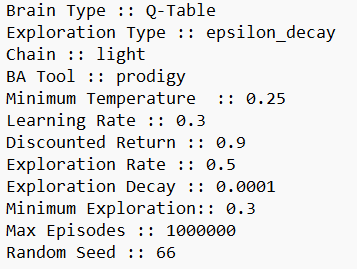
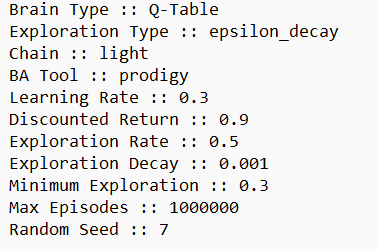


Figure: **Simulation 1 (left):** Used for creating 600 centroids for simulation 3.1.1. **Simulation 2 (Right):** Used for creating 10000 centroids for simulation 3.1.2.

1. The mean score for the clusters was found using the KDTree package of sklearn to find members of clusters and saved along with the centroids.
2. Simulations were run where the Mean Cluster Score replaced the Score as mentioned in **§** 2.3.1.
   * 1. *State Reduction*

Another way of reducing the state space is to restrict the residues we are working with. To perform this, I used data generated by Prof. Subhashini Srinivasan lab member Ms. Apoorva which explains which amino acids are conserved in the different residues in loops of heavy and light chains.

Table: Mutation Bias Matrix for the 17 positions in Pembrolizumab-PD1 (5B8C)

| CDR | Position | A | C | D | E | F | G | H | I | K | L | M | N | P | Q | R | S | T | V | W | Y | Entropy |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CDR-L1 | Ser-32 | 1 | 0 | 4 | 0 | 3 | 1 | 0 | 0 | 0 | 1 | 0 | 6 | 0 | 0 | 0 | 24 | 0 | 0 | 0 | 13 | 1.2747 |
| CDR-L1 | Tyr-34 | 1 | 0 | 7 | 0 | 1 | 1 | 2 | 2 | 5 | 0 | 0 | 8 | 0 | 2 | 0 | 27 | 1 | 1 | 1 | 24 | 1.0824 |
| CDR-L2 | Tyr-53 | 0 | 0 | 0 | 0 | 9 | 0 | 9 | 0 | 11 | 0 | 0 | 1 | 0 | 0 | 1 | 13 | 0 | 0 | 1 | 425 | 2.8889 |
| CDR-L2 | Tyr-57 | 1 | 0 | 3 | 2 | 11 | 3 | 1 | 6 | 4 | 1 | 0 | 106 | 0 | 2 | 15 | 199 | 98 | 2 | 0 | 16 | 1.2454 |
| CDR-H3 | Arg-99 | 19 | 1 | 61 | 22 | 17 | 83 | 12 | 7 | 19 | 18 | 7 | 6 | 31 | 8 | 26 | 33 | 25 | 18 | 12 | 35 | 0.2397 |
| CDR-H3 | Tyr-101 | 11 | 4 | 33 | 16 | 17 | 32 | 6 | 19 | 5 | 13 | 6 | 6 | 4 | 0 | 31 | 21 | 22 | 8 | 5 | 65 | 0.4001 |
| CDR-H3 | Arg-102 | 7 | 4 | 20 | 7 | 11 | 41 | 1 | 2 | 0 | 14 | 5 | 8 | 11 | 1 | 11 | 26 | 23 | 10 | 1 | 70 | 0.5643 |
| CDR-H1 | Thr-30 | 2 | 0 | 0 | 1 | 0 | 4 | 0 | 17 | 6 | 0 | 1 | 23 | 2 | 0 | 9 | 88 | 342 | 0 | 0 | 0 | 1.7489 |
| CDR-H1 | Tyr-33 | 81 | 0 | 13 | 5 | 8 | 40 | 6 | 5 | 1 | 6 | 0 | 21 | 5 | 1 | 1 | 17 | 31 | 31 | 85 | 138 | 1.008 |
| CDR-H1 | Tyr-35 | 1 | 0 | 5 | 7 | 1 | 2 | 236 | 1 | 4 | 0 | 0 | 83 | 0 | 19 | 1 | 99 | 14 | 3 | 0 | 19 | 1.761 |
| CDR-H2 | Asn-52 | 6 | 0 | 64 | 3 | 8 | 0 | 13 | 69 | 13 | 5 | 3 | 194 | 0 | 0 | 1 | 37 | 4 | 15 | 3 | 57 | 1.1057 |
| CDR-H2 | Ser-54 | 6 | 0 | 10 | 19 | 10 | 106 | 9 | 60 | 9 | 9 | 10 | 98 | 0 | 3 | 12 | 63 | 9 | 11 | 0 | 49 | 0.6245 |
| CDR-H2 | Asn-55 | 5 | 0 | 43 | 3 | 49 | 28 | 3 | 5 | 1 | 12 | 0 | 141 | 1 | 2 | 7 | 149 | 27 | 9 | 2 | 8 | 0.9189 |
| CDR-H2 | Thr-58 | 70 | 0 | 0 | 1 | 0 | 4 | 0 | 7 | 3 | 0 | 1 | 5 | 17 | 0 | 0 | 13 | 366 | 7 | 0 | 0 | 1.8005 |
| CDR-H2 | Asn-59 | 12 | 0 | 17 | 21 | 3 | 19 | 13 | 12 | 68 | 0 | 7 | 211 | 1 | 6 | 21 | 51 | 19 | 4 | 0 | 10 | 0.9348 |
| CDR-L3 | Ser-95 | 12 | 1 | 3 | 3 | 12 | 13 | 10 | 3 | 1 | 15 | 3 | 11 | 1 | 1 | 54 | 88 | 1 | 0 | 10 | 222 | 1.374 |
| CDR-L3 | Asp-97 | 4 | 0 | 28 | 11 | 2 | 13 | 2 | 6 | 5 | 3 | 4 | 83 | 1 | 5 | 10 | 179 | 30 | 4 | 4 | 6 | 0.9694 |

The mutation bias matrix was used to reduce the state space. In total there were different sets of reduced states which were tested in the simulations:

1. Light Chain: Using all 6 positions of light chain but mutating them only into:
   1. 3 amino acids:
      1. S, G and D: For Simulation 3.2.1.1
      2. S, Y and N: For Simulation 3.2.1.2
   2. 6 amino acids: S, Y, N, R, T and D for Simulation 3.2.2.1
2. Heavy Chain: Only Mutating selected positions for the mutations and mutating them into other residues
   1. Mutating 3 residues (CDR-H1: Tyr-33, CDR-H2: Ser-54, CDR-H3: Arg-99) into 17 a.a. ('A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y').
   2. Mutating 4 residues (CDR-H2: Ser-54, CDR-H3: Arg-99, Tyr-101 and Arg-102) into 16 a.a. ('A', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'N', 'R', 'S', 'T', 'V', 'W', 'Y')
   3. **Deep Q-Learning**

The basic pipeline for Deep Q-Learning is adapted from Mnih V et al., 2013. Pipeline was made using PyTorch (v.2.2.1+cu121):

1. Initialize the Brain class with hyper-parameters like exploration rate, learning rate, network sync. Rate, replay memory size, memory mini batch size etc. and initialize the replay memory.
2. Initialize the Policy network with random weights:
   1. Using FFNN with following layers:
      1. Input layer: 340🡪340; input was created by doing one hot coding of 17 positions and 20 a.a. creating a binary tensor of size 340
      2. Hidden layers: 3 hidden layers of: 340🡪170, 170🡪85 and 85🡪42
      3. Output layer: 42🡪size of action list; there were 51 actions for the simulation I ran (3 positions 🡪 17 a.a.).
3. Clone the Policy network into target network
4. Loop for the Episodes:
   1. Select an action using policy network with Epsilon greedy strategy
   2. Execute action in PyMOL and save state PDB in the local PDB bank
   3. Observe State, Reward and next state
   4. Store experience tuple (state, action, reward, next state) in replay memory
   5. Sample random mini batch from replay memory
   6. Pass replay mini batch to policy network
   7. Calculate loss (MSE) b/w output Q values (from policy network) and target Q value
      1. Requires passing states through target network (clone of policy network, updated after certain episodes)
      2. Target Q value is calculated using the Q-value update equation but using target network output as replacement for Q-values.
   8. Gradient Descent (Optimizer: Adam) updates weight in policy network to minimize loss
   9. After certain episodes (network sync rate), weights in the target network are updated with the policy network.

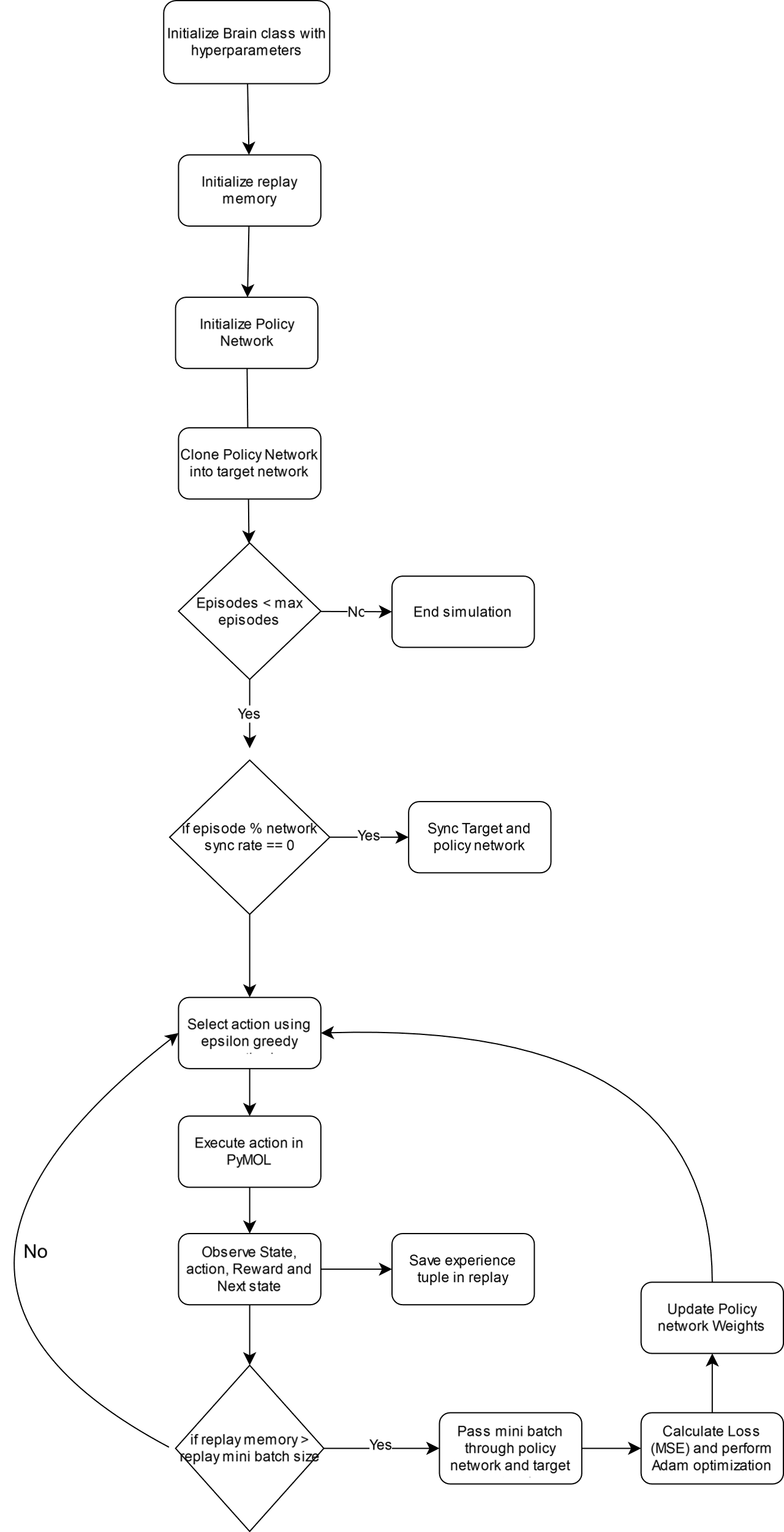


Figure: Flowchart of DQL pipeline

* 1. **Structure Validation**
     1. *AlphaFold2 and RMSD*

We have used LocalColabFold 1.5.0 which uses AlphaFold 2.3.1 for predicting the states. This allows us to perform validation on if the state is easily folding following the assumptions mentioned in **§** 1.1.1.

The pipeline:

1. The last 10000 episodes of the simulation is selected and sorted in ascending order of BA
2. The PDB of states are extracted from local PDB bank
3. The PDB are converted into FASTA format (LocalColabFold format for multimer) using pdb2fasta (<https://zhanggroup.org/pdb2fasta/pdb2fasta>)

> State

{sequence of chain A}:{sequence of chain B}:{sequence of chain C}

1. LocalColabFold is used to predict the structures. The command used:

$ nohup colabfold\_batch --amber --use-gpu-relax --num-recycle 20 --num-models 5 --num-relax 1 --recycle-early-stop-tolerance 0.01 {file\_name}.fasta {file\_name} > {file\_name}.log &

1. The relaxed structures are separated and PRODIGY is run to get BA
2. RMSD is calculated of the predicted structure against the Pembrolizumab structure used as initial state in simulations using PyMOL. Commands used:

load {pembrolizumab control}.pdb

load {AF predicted structure}.pdb

align {pembrolizumab control}, {AF predicted structure}

rms\_cur {pembrolizumab control}, {AF predicted structure}, matchmaker=1

* + 1. *C-alpha – C-alpha distance plot*

The C-alpha to C-alpha distance of the PDB were calculated using PDB class from BioPython (v. 1.80) for chain A vs chain A, chain B vs chain B, chain A vs chain B, chain A vs chain C, chain B vs chain C and heatmap was created for the distance matrices.

1. **RESULTS AND DISCUSSIONS (max 4000 words)**
   1. **State Aggregation to Reduce State Space**
      1. *Simulation 1*

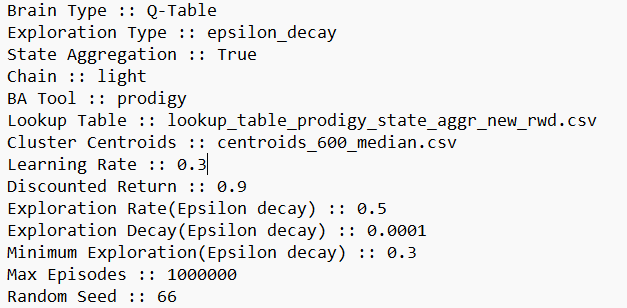


Figure: Hyperparameters for simulation

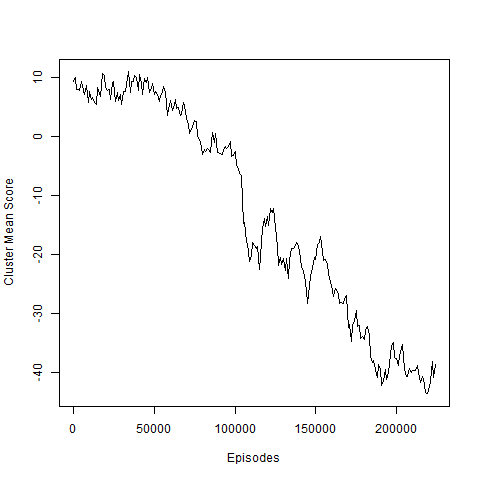
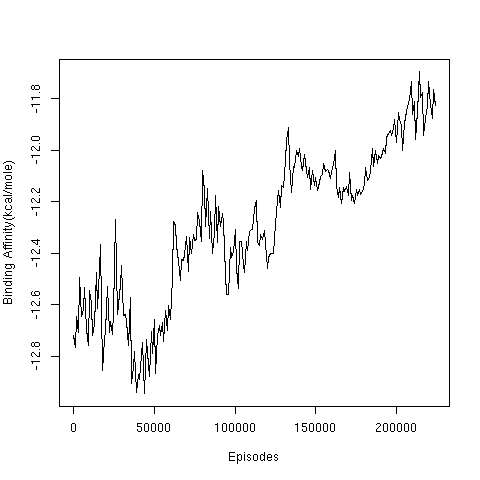


Figure: Average BA (left) and Cluster Mean Score (Right) per 1000 episodes

We can observe in the graphs that the BA increased and the score decreased throughout the simulation. This shows that the clustering method we used didn’t help. Here we choose 600 centroids which is too less for a state space of 1720, thus increasing the number of clusters may help. So, in the next simulation 10 000 clusters are used.

* + 1. *Simulation 2*

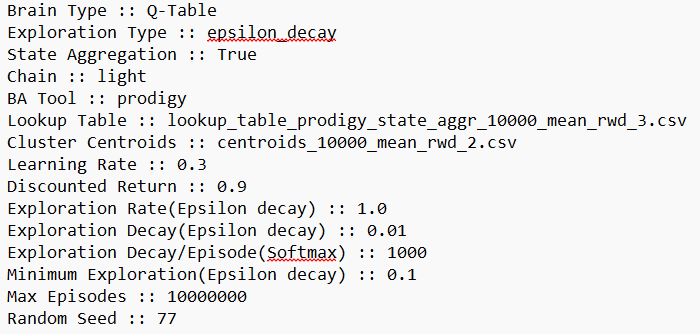


Figure: Hyperparameters for simulation

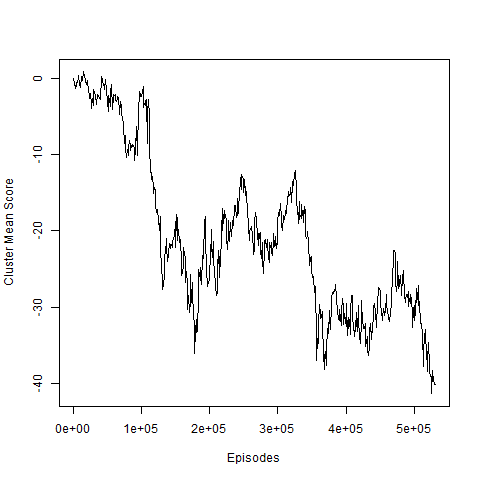
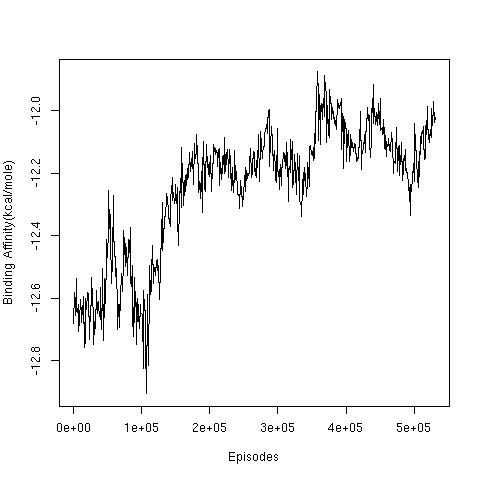


Figure: Average BA (left) and Cluster Mean Score (Right) per 1000 episodes

We observed that the BA increased and score reduced in the simulation again. This shows that state aggregation didn’t work. This could be because of various reasons:

* The hyperparameters and scoring function were incorrect and need optimization
* The state space is still too big for us to aggregate them into 10 000 clusters, thus clustering method used is not useful for our state space, perhaps a different method could work
* We used different types of contacts to cluster the states, but due to it being a big state space and different sequences in states can be clustered into the same cluster, but when taking action on these states they randomly jump into different clusters. This is because the action is based on the sequence and clustering based on interactions. A clustering method based on a.a. sequence may be much more appropriate for our use case.
  1. **Mutating residues of Light Chain**
     1. *Mutating residues of light chain into 3 residues*
        1. *Simulation 1*

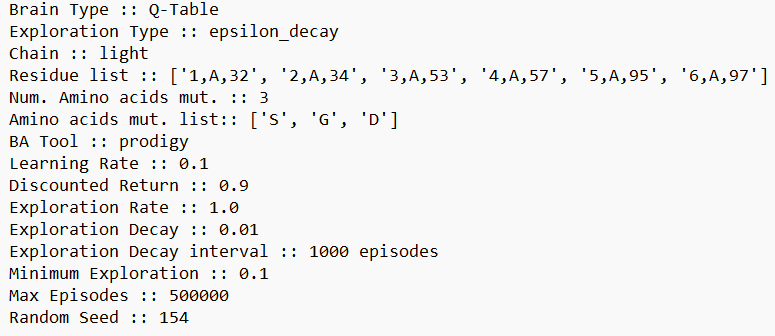


Figure: Hyperparameters for simulation

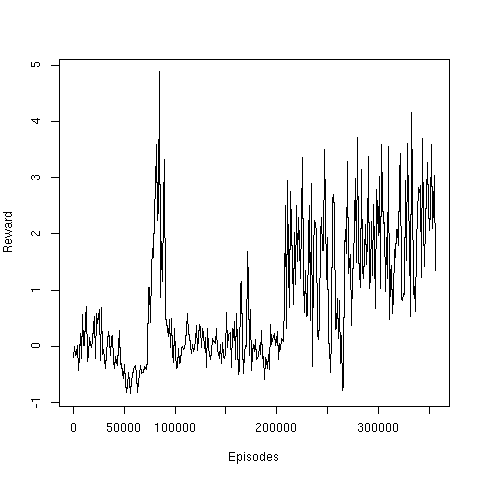
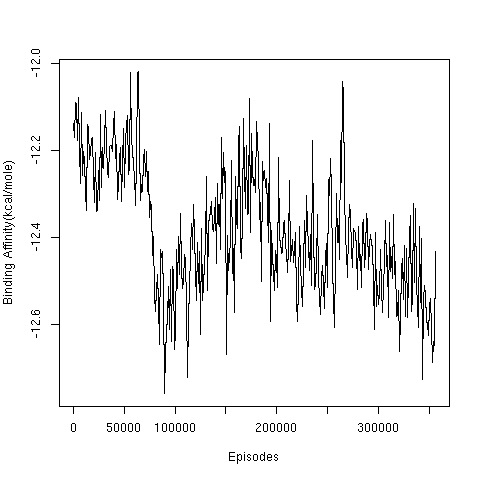


Figure: Average BA (left) and Reward (Right) per 1000 episodes

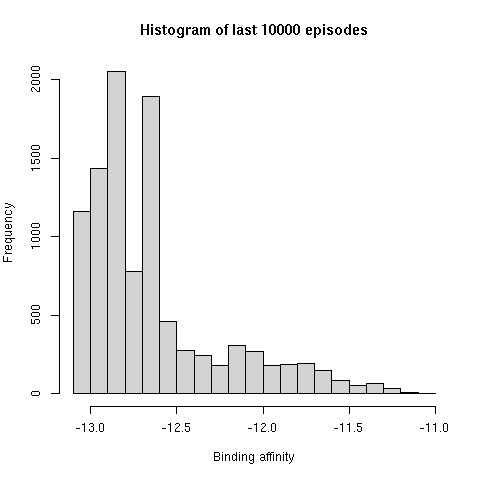
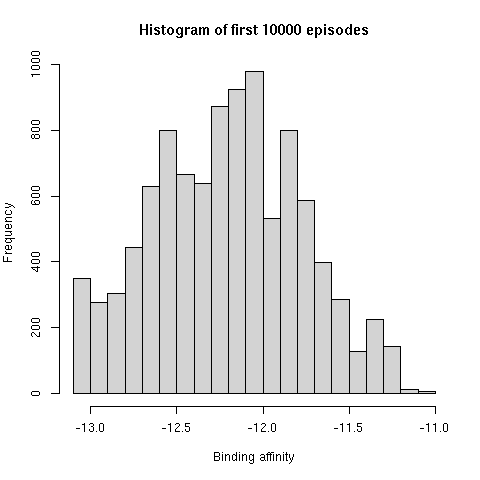


Figure: Histogram of first (left) and last (right) 10 000 episodes of simulation

We have used relatively small a.a. (S, G and D) for substitution in this simulation. We observed that the BA decreased and Reward increased over time. This is especially observable in the histogram where the frequency of better BA states has increased in the last 10 000 episodes as compared to the first 10 000 episodes. Here the van der Waals negative reward was -0.5.

* + - 1. *Simulation 2*

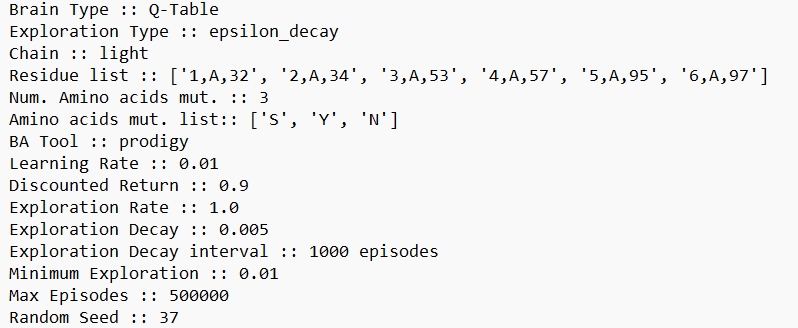


Figure: Hyperparameters for simulation

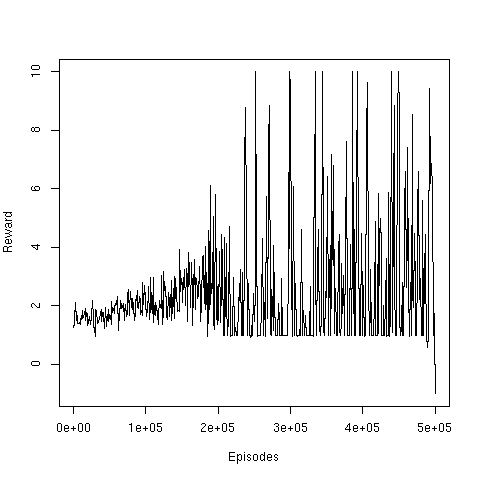
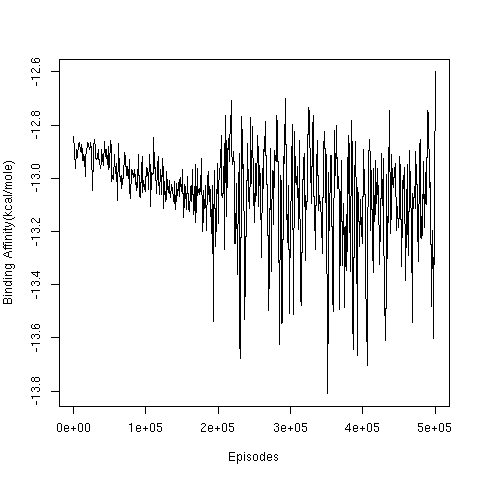


Figure: Average BA (left) and Reward (Right) per 1000 episodes

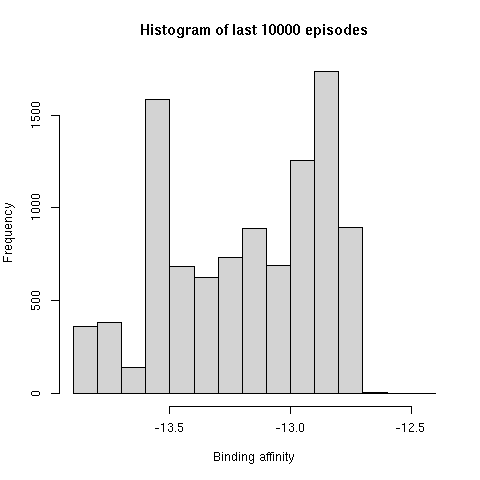
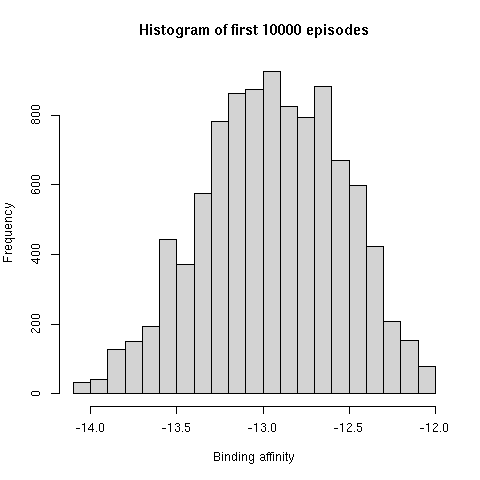


Figure: Histogram of first (left) and last (right) 10 000 episodes of simulation

For this simulation relatively larger a.a. (S, Y and N) were substituted as compared to before, these were conserved a.a. observed in the mutation bias matrix. We observed that till 2e+5 episodes the BA decreased and Score increased, then it started to fluctuate highly as more states with +10 rewards are observed. This simulation also has a mean BA lower than the simulation 1 observed in the first 10 000 episodes, signifying that these mutations are preferred more than those selected in simulation 1.

* + 1. *Mutating residues of light chain into 6 residues* 
       1. *Simulation 1*

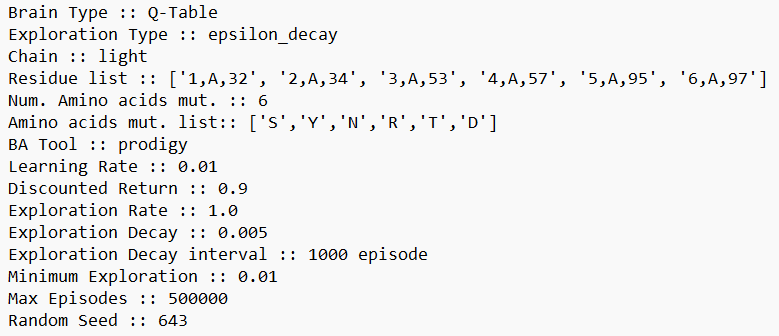


Figure: Hyperparameters for simulation

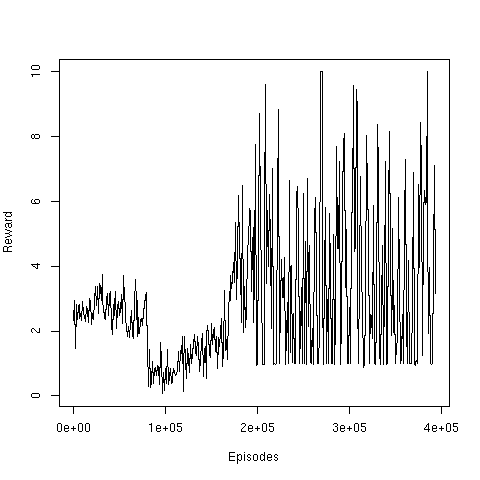
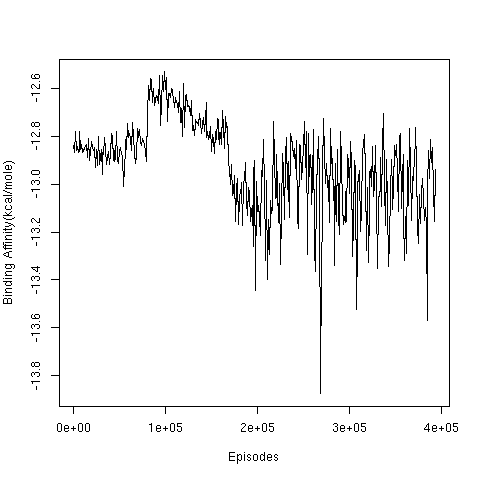


Figure: Average BA (left) and Reward (Right) per 1000 episodes

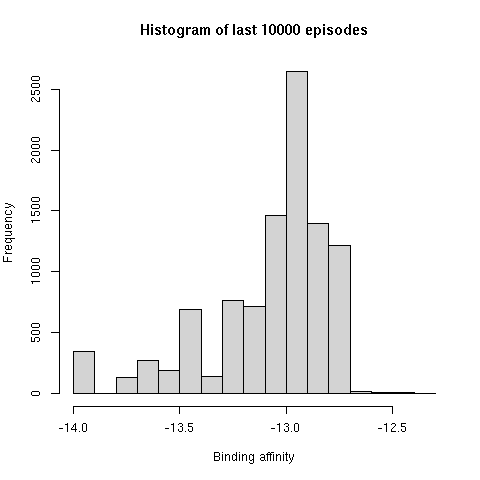
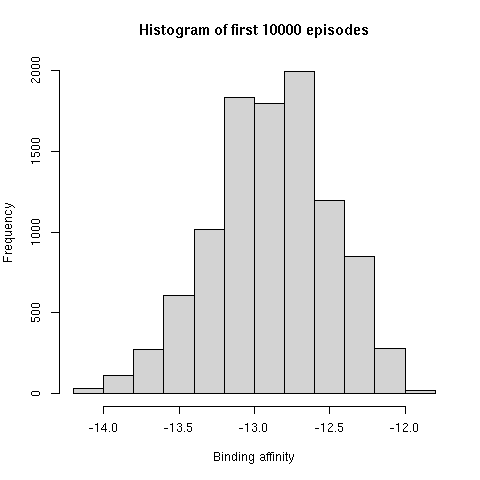


Figure: Histogram of first (left) and last (right) 10 000 episodes of simulation

Here we increased the state space by mutating light chain residues into 6 a.a., these were also conserved a.a. observed in the light chain and we can observe that the results were similar to the simulation 2 with S, Y and N. Although overall fluctuation on BA is less compared to simulation 2. We observed states with better BA being preferred at the end of simulation comparing the histogram of first and last 10 000 episodes.

We selected conserved amino acids for substitution in the light chain, this does give us better binding states, but to create a more biological model for SHM, it is important to explore positions which can tolerate more types of variations. So we did simulations based on less conserved positions on the heavy chain next.

* 1. **Mutations on Heavy Chain**
     1. *Mutating 3 residues into 17 residues*
        1. *Simulation 1*

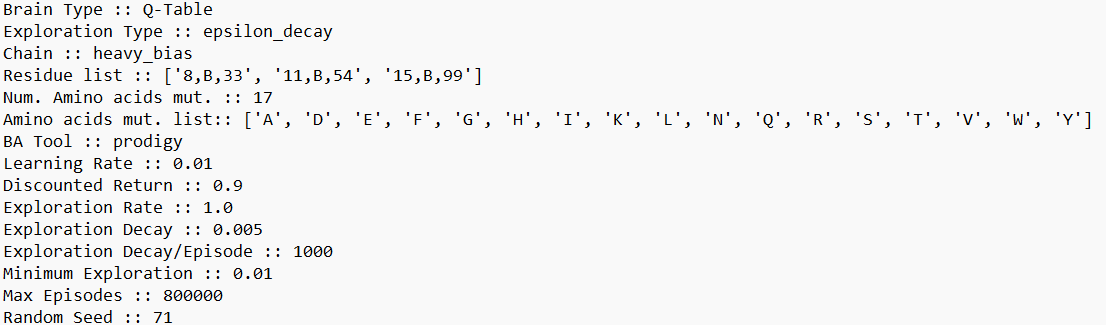
**

Figure: Hyperparameters for simulation

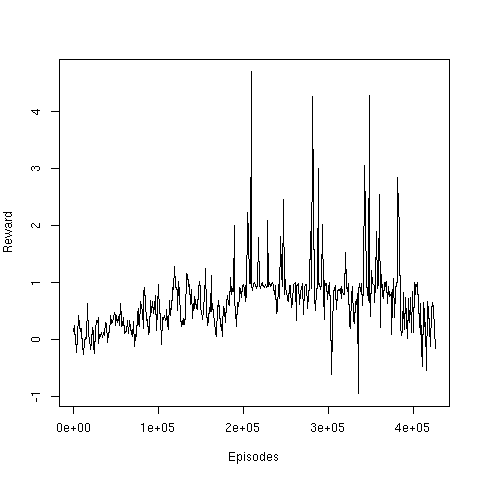
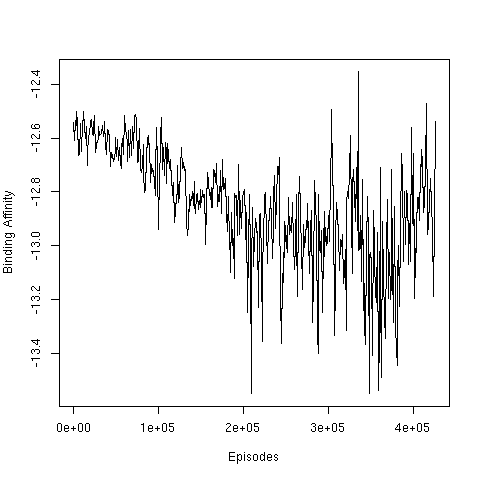


Figure: Average BA (left) and Reward (Right) per 1000 episodes

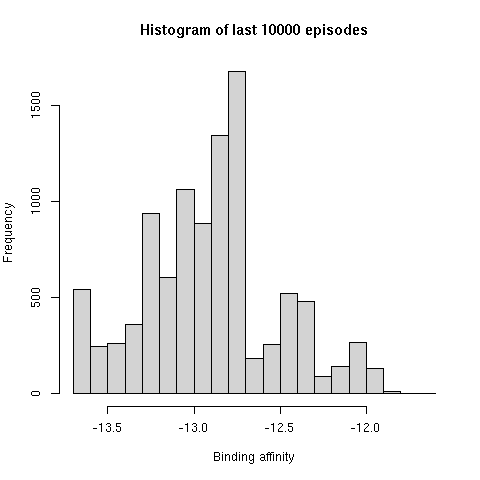
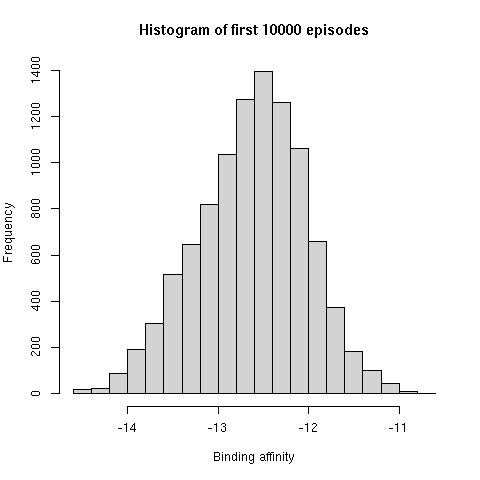


Figure: Histogram of first (left) and last (right) 10000 episodes of simulation

We can observe that the overall BA decreased and score increased by episodes. But here the mean of the first 10 000 episodes is around -12.6 kcal/mole which is the same as our initial state. In previous simulations on light chains, the mean was either higher or less than this, this simulation thus seems more natural for our stated state. Also, the variation in the BA and Reward has reduced significantly as compared to simulations on light chains. The overall frequency of better BA states increased significantly compared to the first 10 000 episodes. To make sure this is not a fluke, let us use a different random seed for the same parameters.

* + - 1. *Simulation 2*

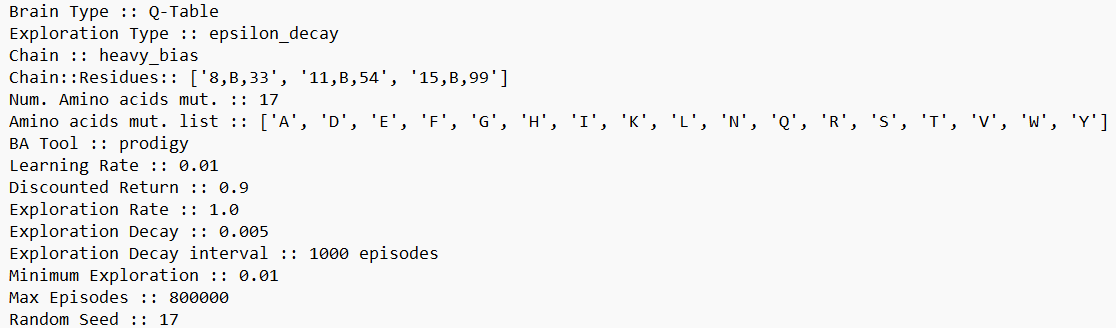


Figure: Hyperparameters for simulation

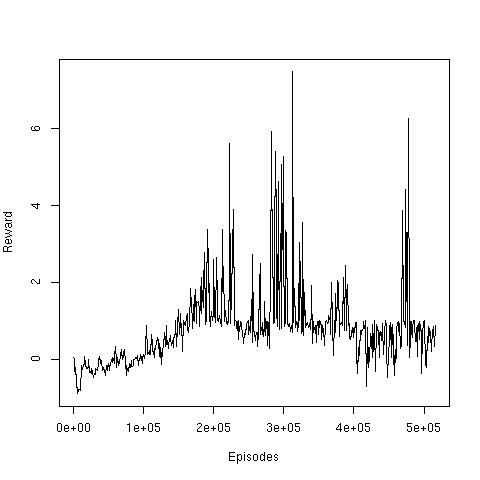
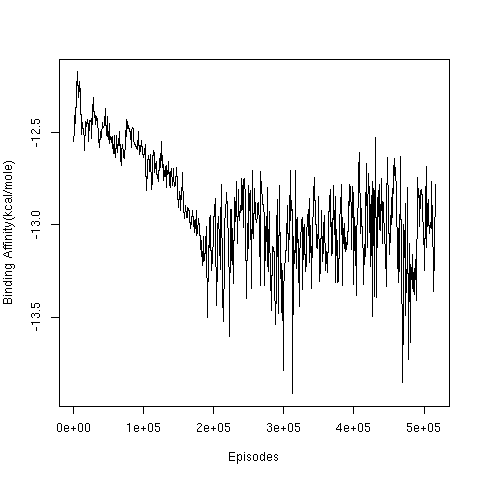


Figure: Average BA (left) and Reward (Right) per 1000 episodes

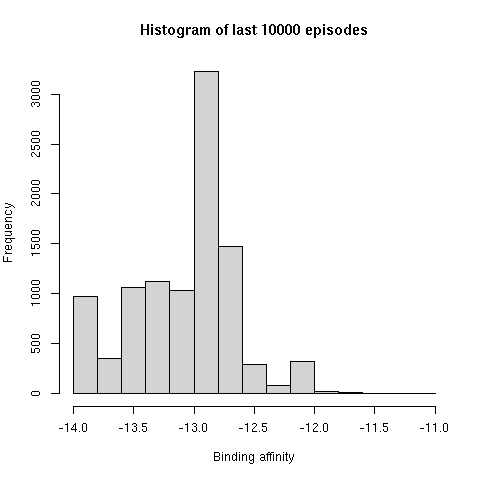
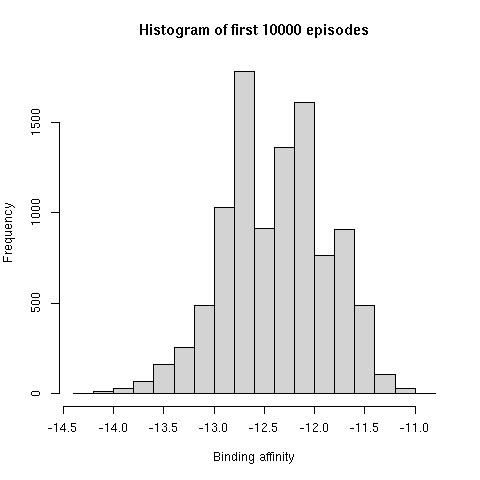


Figure: Histogram of first (left) and last (right) 10000 episodes of simulation

We can observe that even after changing the random seed the simulation showed similar results to the earlier simulation. Let us try a different exploration method i.e. soft-max for these positions.

* + - 1. *Simulation 3*

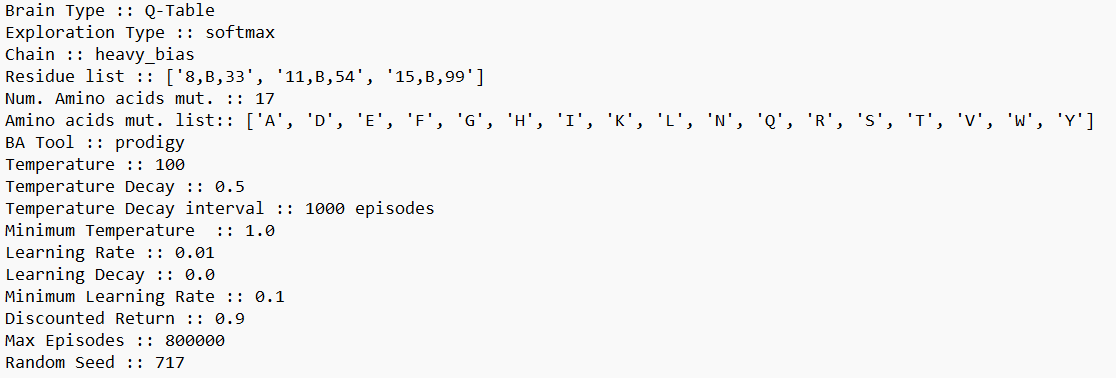


Figure: Hyperparameters for simulation

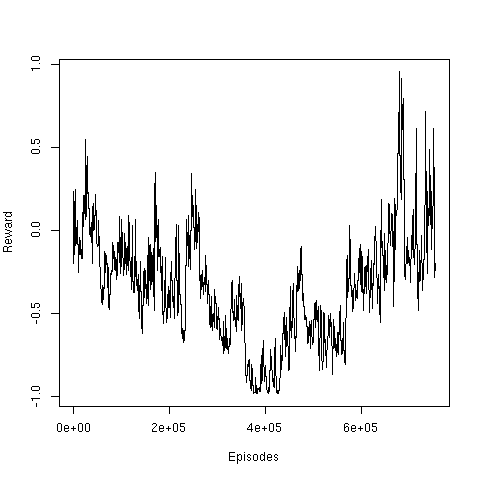
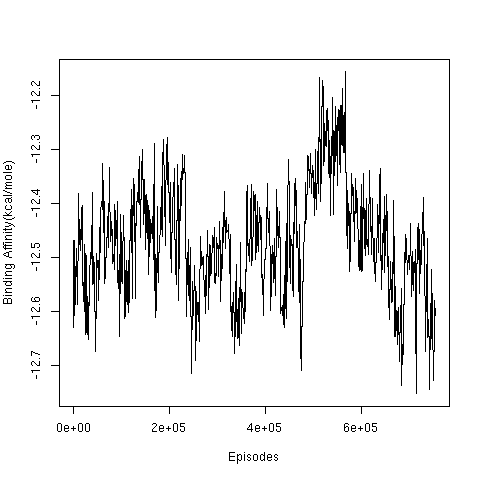


Figure: Average BA (left) and Reward (Right) per 1000 episodes

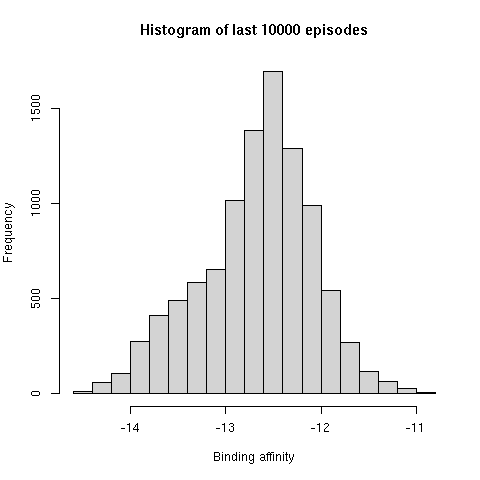
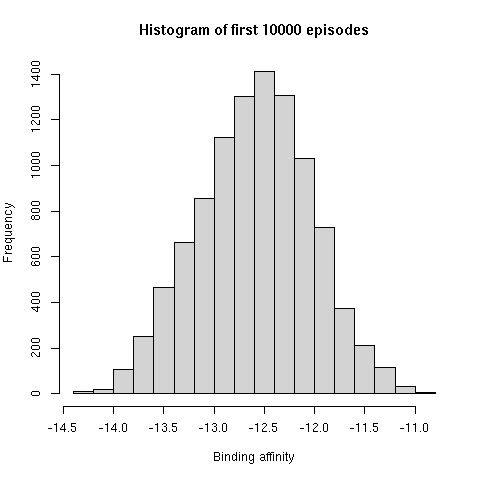


Figure: Histogram of first (left) and last (right) 10000 episodes of simulation

We can observe that the BA and score remained very random throughout the simulation. The histogram of the first and last 10 000 states are also very similar. This can be because the parameters for soft-max are not optimized, when comparing with epsilon greedy. We then tried to use a different set of positions in the heavy chain, these four positions had the least entropy as we can observe in the mutation bias matrix.

* + 1. *Mutating 4 residues into 16 residues*
       1. *Simulation 1*

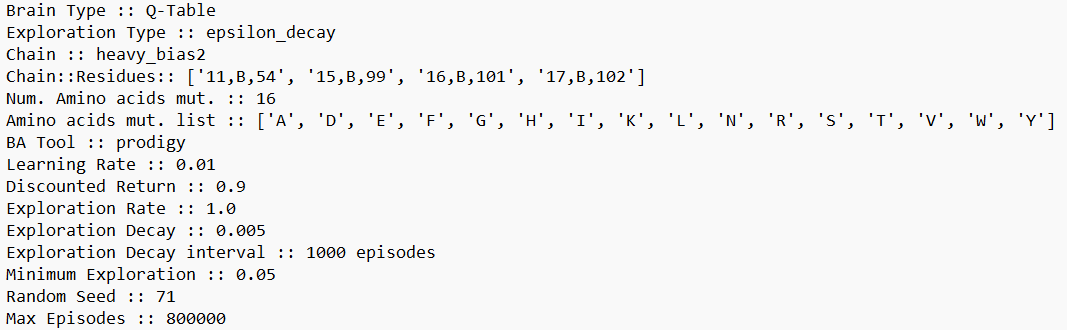


Figure: Hyperparameters for simulation

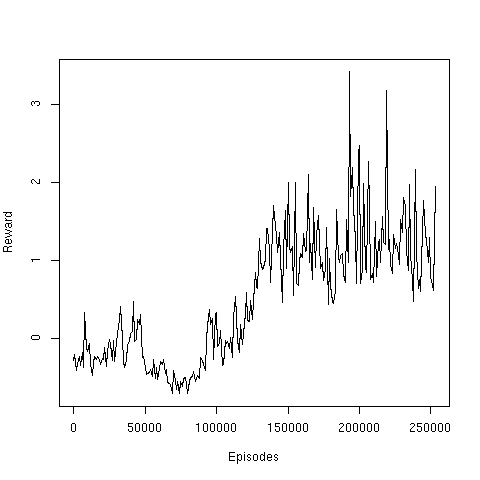
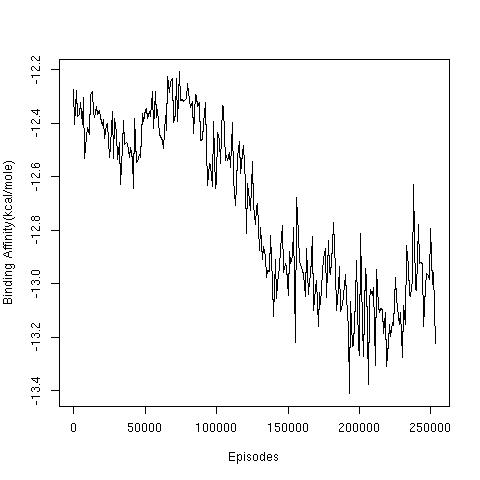


Figure: Average BA (left) and Reward (Right) per 1000 episodes

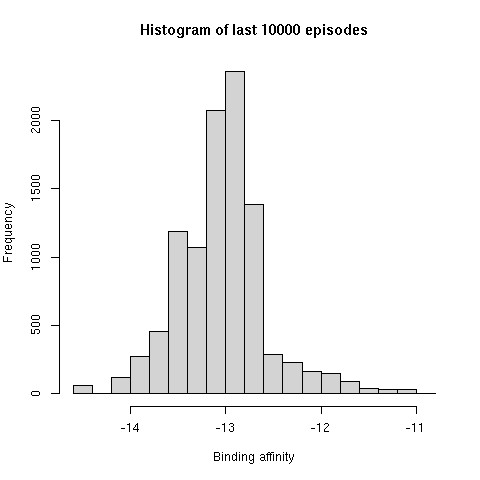
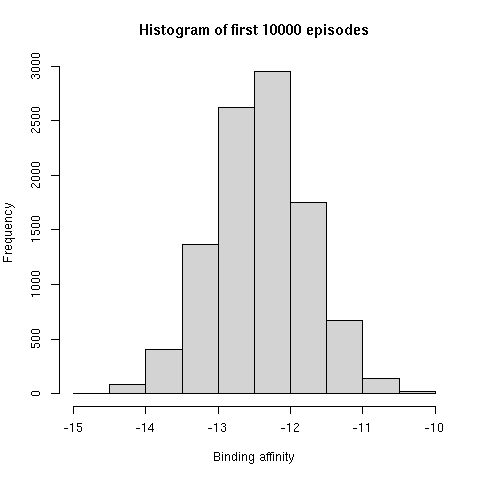


Figure: Histogram of first (left) and last (right) 10000 episodes of simulation

We observed that the BA decreased and Reward increased drastically around the 1.5e+5 episodes. The variation in the BA and Reward also seems to be reduced when compared to simulations before. This simulation also provided higher binding affinity states.

We tried a different algorithm for the next simulation to try to replicate the biological system in which the states where the B cells with high BA are killed by the system, thus here if a state reaches -10 reward, the simulation resets to the initial pembrolizumab state.

* + - 1. *Simulation 2*

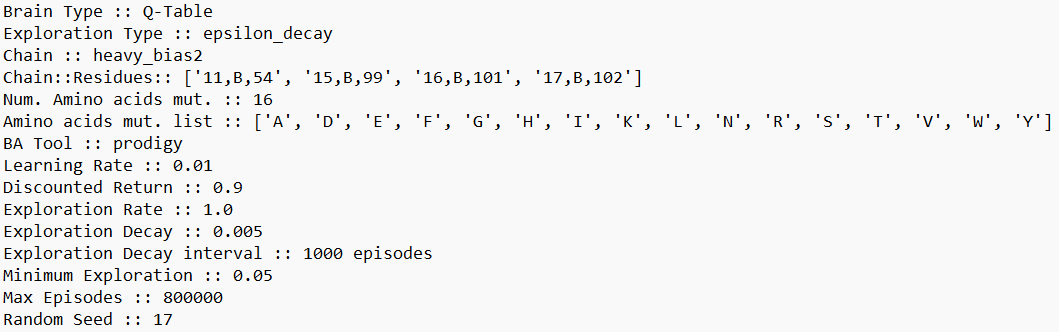


Figure: Hyperparameters for simulation

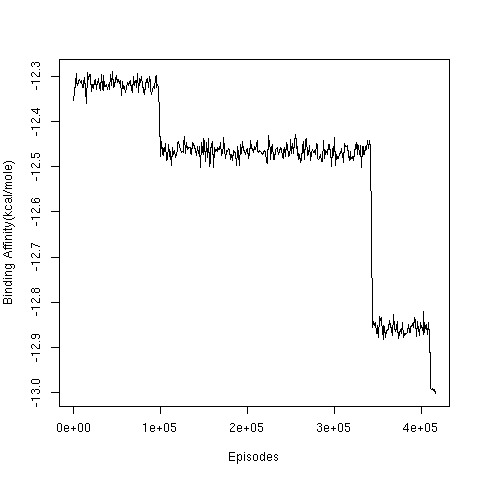
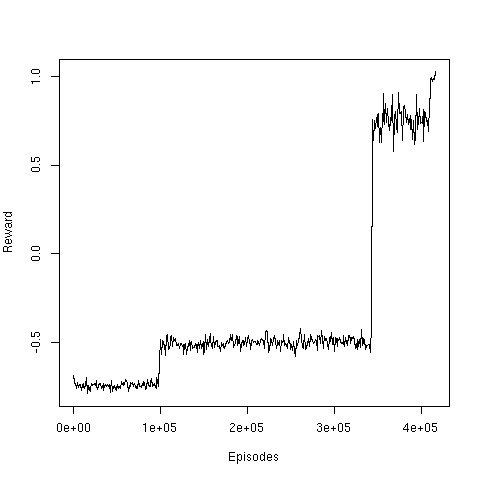
 

Figure: Average BA (left) and Reward (Right) per 1000 episodes

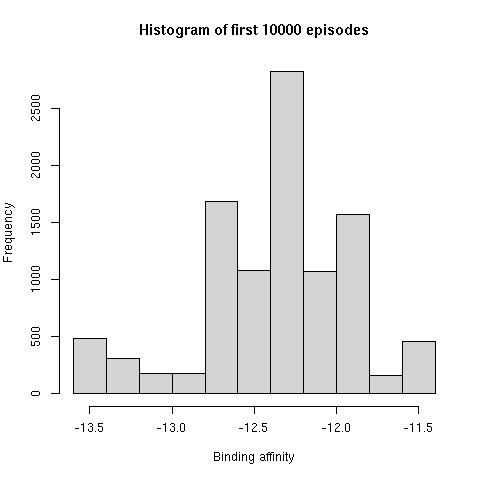
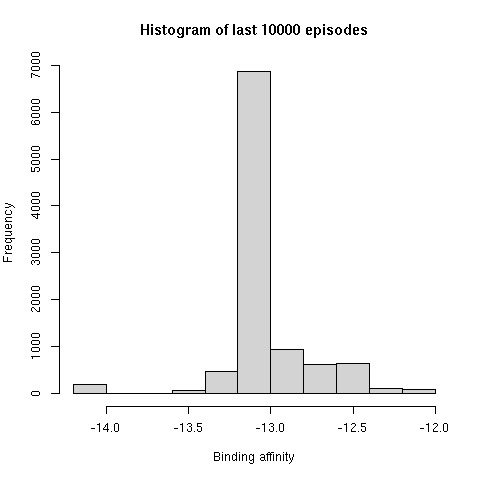
 

Figure: Histogram of first (left) and last (right) 10000 episodes of simulation

What an interesting change in the flow of the simulation. As compared to before, here the reduction in the BA and increase in Reward is much more discrete and in steps. And the last 10 000 are dominated by a very limited range of states, although we didn’t perform the structure validation on these states yet. It seems that the new condition of survival of states restricts the exploration also, as now the agent cannot explore states which might need to jump though few -10 reward states.

* 1. **Deep Q Learning**
     1. *Simulation 1*

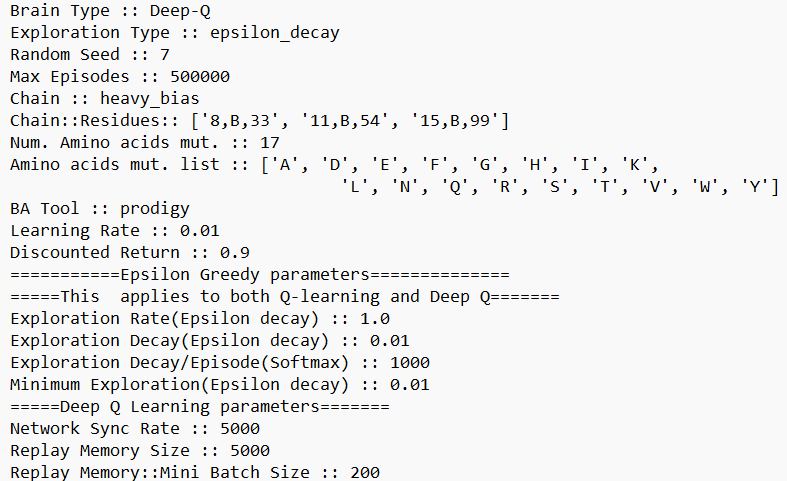
****

Figure: Hyperparameters for simulation

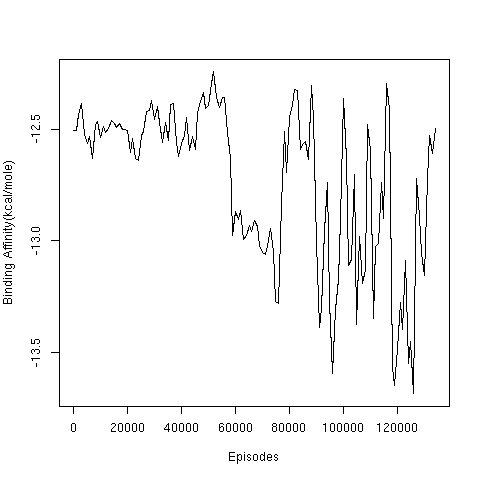
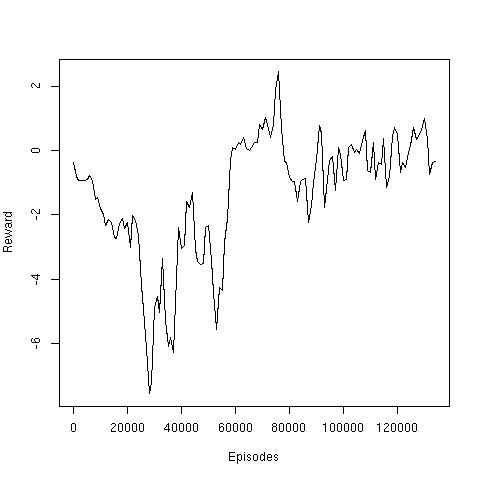


Figure: Average BA (left) and Reward (Right) per 1000 episodes

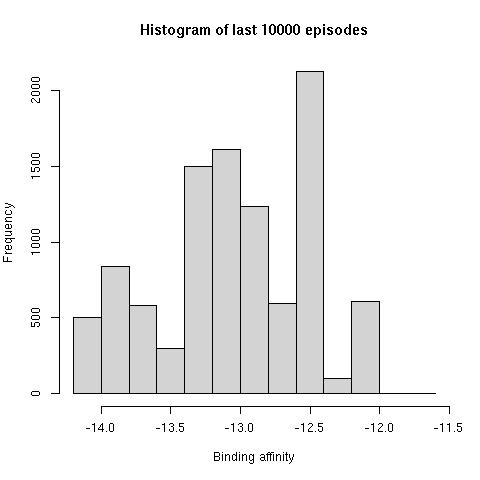
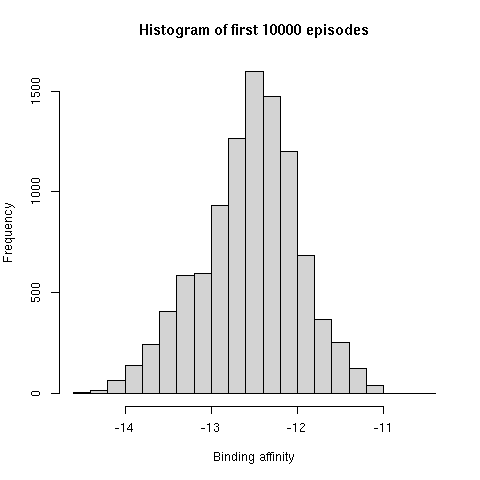


Figure: Histogram of first (left) and last (right) 10000 episodes of simulation

The DQL simulation is much slower and more compute intensive as compared to Q-Learning, given the time constraint, only one simulation was possible. We can observe that the simulation is very random. Although we got some good states from the simulation, but the simulation itself requires optimization of parameters. The input tensor also can be changed from a simple one hot coding of the state to include factors such as type of a.a. in the positions i.e. hydrophobic, charged, polar, apolar, etc. and the types of interactions each position have with PD1. This needs much more time and study to create a good simulation.

* 1. **Validation of Higher Affinity States**
     1. *Validation by AlphaFold2 and RMSD*
        1. *States of mutations on light chain*

Table: Simulation 1 of mutating 6 residues of light chain into 3 a.a.

| State | Freq | Reward | Crystal BA | AF BA | AF RMSD |
| --- | --- | --- | --- | --- | --- |
| DDDSGDTYYNSNTNRYR | 55 | 10 | -13.1 | -13.4 | 2.264 |
| GDSSGGTYYNSNTNRYR | 46 | 10 | -12.9 | -13.3 | 2.296 |
| SSSSGDTYYNSNTNRYR | 92 | 10 | -13.0 | -13.3 | 2.129 |
| SDSSGGTYYNSNTNRYR | 48 | 10 | -12.7 | -13.2 | 2.478 |
| GSDSGDTYYNSNTNRYR | 62 | 10 | -13.1 | -13.0 | 2.238 |

Table: Simulation 1 of mutating 6 residues of light chain into 6 a.a.

| State | Freq | Reward | Crystal BA | AF BA | RMSD |
| --- | --- | --- | --- | --- | --- |
| DRRTSYTYYNSNTNRYR | 65 | 10 | -13.4 | -23.3 | 19.48 |
| STRNYYTYYNSNTNRYR | 45 | 10 | -13.6 | -13.5 | 2.061 |
| YRNNSRTYYNSNTNRYR | 141 | 10 | -13.4 | -13.4 | 2.058 |
| DSRNYYTYYNSNTNRYR | 41 | 10 | -13.7 | -13.3 | 2.072 |
| DTRNYYTYYNSNTNRYR | 89 | 10 | -13.7 | -13.3 | 1.987 |
| YRNNNRTYYNSNTNRYR | 101 | 10 | -13.3 | -13.3 | 2.072 |

We have only shown the top 5 states with respect to Reward and binding affinity after AlphaFold2. We can observe that AlphaFold2 can incorrectly fold the antibodies for certain mutations. This is explained in detail later.

* + - 1. *States of mutations on heavy chain*

Table: Simulation 1 of mutating 3 residues of heavy chain into 17 a.a.

| State | Freq | Reward | Crystal BA | AF BA | AF RMSD |
| --- | --- | --- | --- | --- | --- |
| SYYYSDTWYNYNTNGYR | 21 | 1 | -13.0 | -25.3 | 20.214 |
| SYYYSDTRYNLNTNYYR | 246 | 1 | -13.1 | -13.9 | 2.205 |
| SYYYSDTRYNLNTNDYR | 72 | 1 | -13.3 | -13.6 | 2.205 |
| SYYYSDTDYNLNTNDYR | 90 | 1 | -12.8 | -13.5 | 2.161 |
| SYYYSDTLYNLNTNEYR | 61 | 1 | -13.5 | -13.5 | 2.121 |
| SYYYSDTRYNFNTNWYR | 81 | 10 | -14.0 | -13.3 | 2.176 |

Table: Simulation 2 of mutating 3 residues of heavy chain into 17 a.a.

| State | Freq | Reward | Crystal BA | AF BA | RMSD |
| --- | --- | --- | --- | --- | --- |
| SYYYSDTDYNFNTNYYR | 19 | 1 | -13.4 | -13.4 | 2.140 |
| SYYYSDTHYNYNTNYYR | 215 | 1 | -13.0 | -13.4 | 2.196 |
| SYYYSDTIYNLNTNVYR | 132 | 1 | -13.3 | -13.3 | 2.135 |
| SYYYSDTEYNYNTNFYR | 93 | 1 | -13.2 | -13.2 | 2.244 |
| SYYYSDTNYNWNTNIYR | 41 | 1 | -12.9 | -13.2 | 2.197 |

Table: Simulation 2 of mutating 4 residues of heavy chain into 16 a.a.

| State | Freq | Reward | Crystal BA | AF BA | RMSD |
| --- | --- | --- | --- | --- | --- |
| SYYYSDTYYNWNTNTHL | 60 | 1 | -12.9 | -13.5 | 2.195 |
| SYYYSDTYYNYNTNIFN | 53 | 1 | -13.0 | -13.5 | 2.172 |
| SYYYSDTYYNYNTNWAK | 143 | 1 | -13.0 | -13.4 | 2.101 |
| SYYYSDTYYNFNTNAWY | 45 | 10 | -14.0 | -13.3 | 2.217 |
| SYYYSDTYYNFNTNIYY | 39 | 10 | -14.1 | -13.0 | 2.185 |

As compared to the light chain mutations, less states with +10 rewards are observed towards the end of simulation. This shows that even though there are less positions on the light chain we are working with the set of 3 a.a. and 6 a.a. affect the BA of antibodies to a great extent.

* + - 1. *States from DQL*

Table: Simulation 1 of DQL

| State | Freq | Reward | Crystal BA | AF BA | RMSD |
| --- | --- | --- | --- | --- | --- |
| SYYYSDTFYNWNTNHYR | 342 | 1 | -13.3 | -13.2 | 2.157 |
| SYYYSDTKYNVNTNTYR | 21 | 1 | -12.7 | -12.9 | 2.218 |
| SYYYSDTYYNWNTNGYR | 112 | 1 | -13.5 | -12.8 | 2.275 |

We were able to get higher BA states for the episodes we ran our simulation, although this doesn’t satisfy our major objective for the project, which is to correctly simulate SHM.

* + 1. *Validation by C-alpha plot*
       1. *Pembrolizumab (control)*

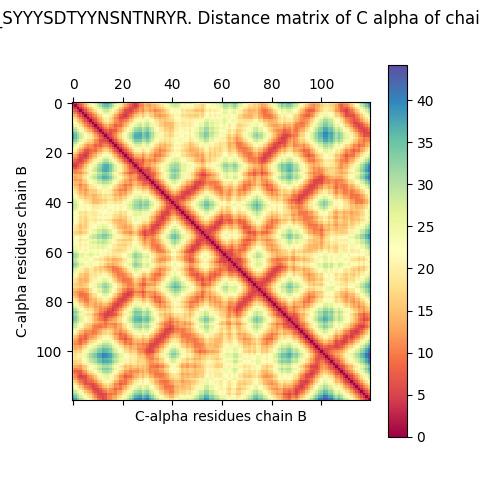
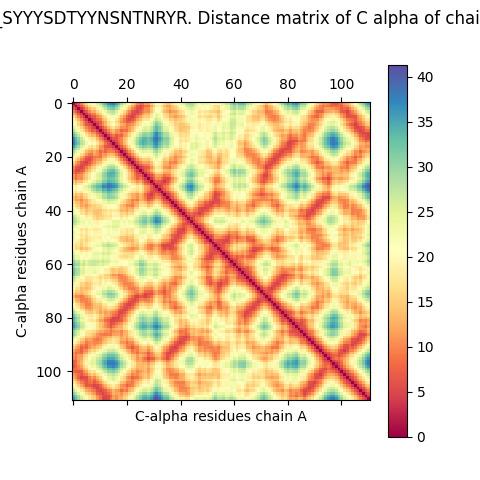


Figure: C-alpha distance plot of chain A vs chain A and chain B vs chain B

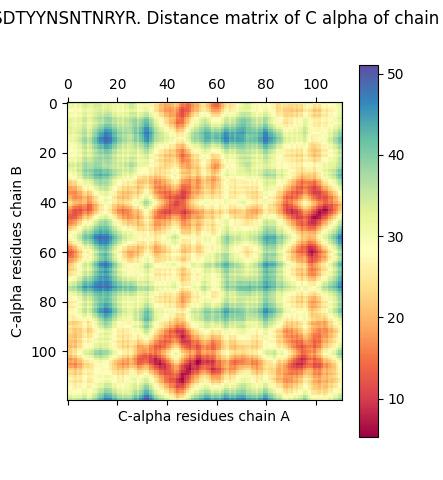


Figure: C-alpha distance plot of chain A vs chain B

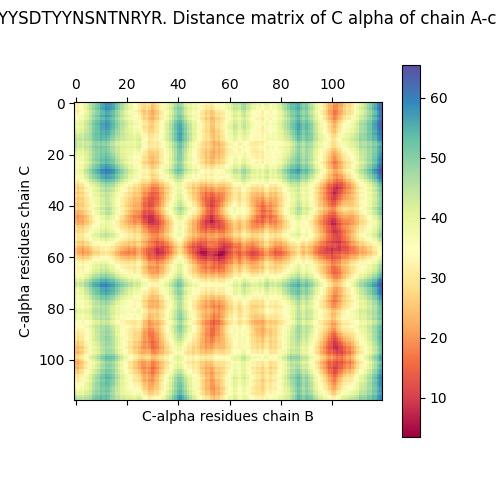
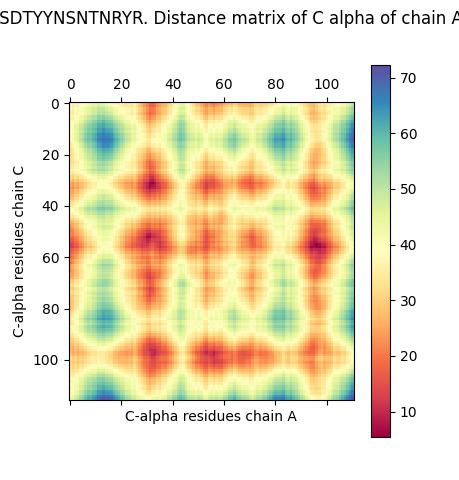


Figure: C-alpha distance plot of chain A vs chain C and chain B vs chain C

These plots are used to validate if our assumption about SHM model in **§** 1.1.1 holds for the states produced by our model and AphaFold2.

* + - 1. *Properly folded state after AlphaFold2 prediction*

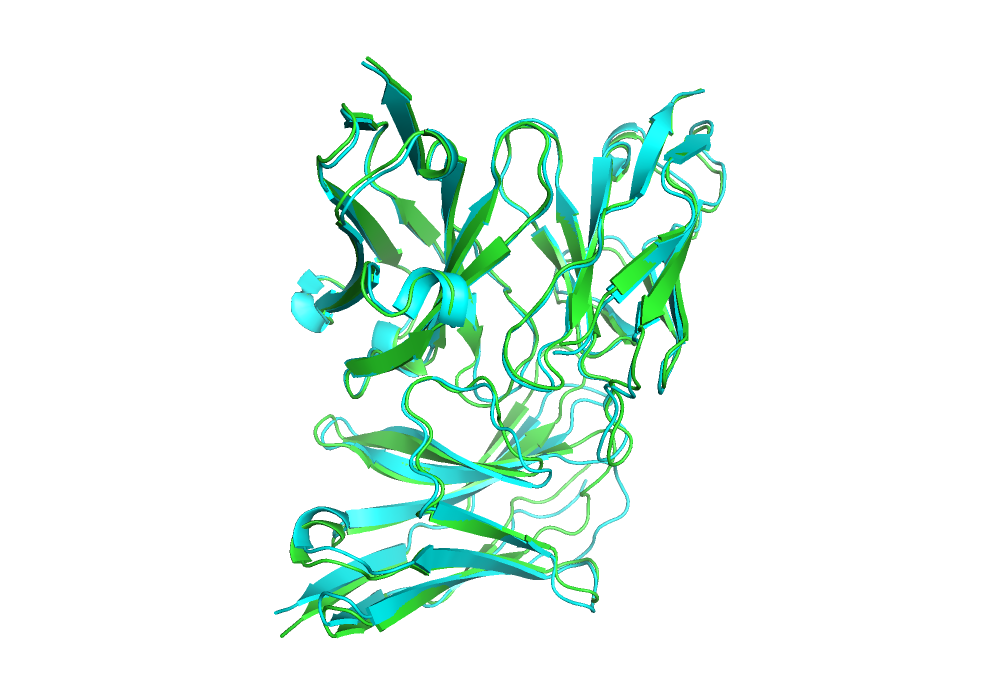


Figure: AF predicted structure of state DDDSGDTYYNSNTNRYR aligned with pembrolizumab (control) state

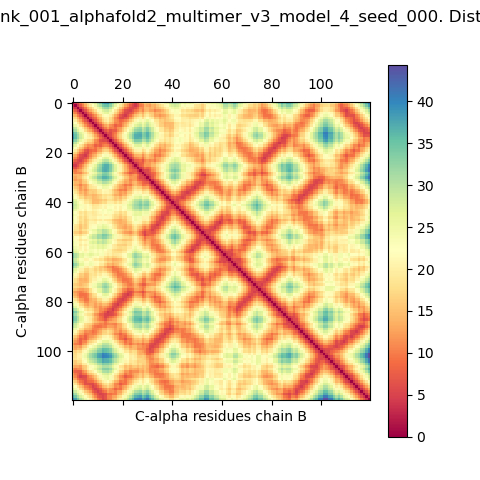
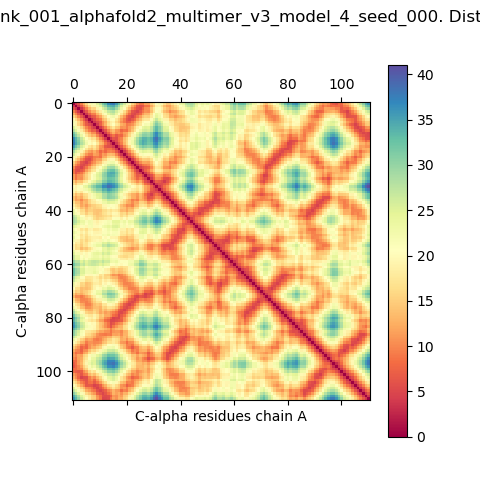


Figure: C-alpha distance plot of chain A vs chain A and chain B vs chain B

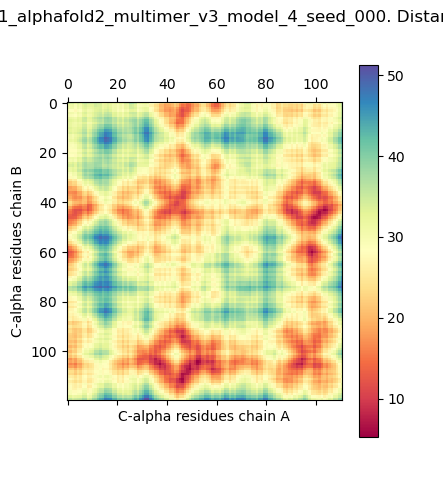


Figure: C-alpha distance plot of chain A vs chain B

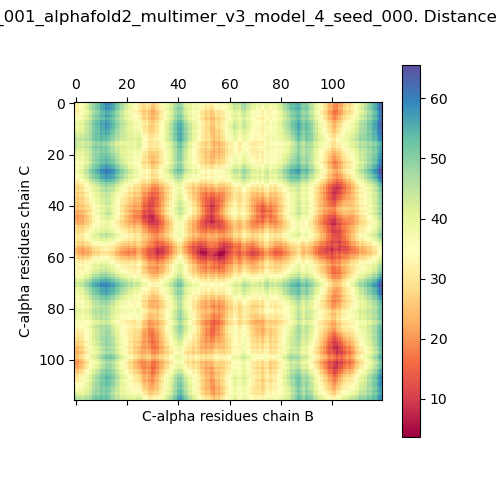
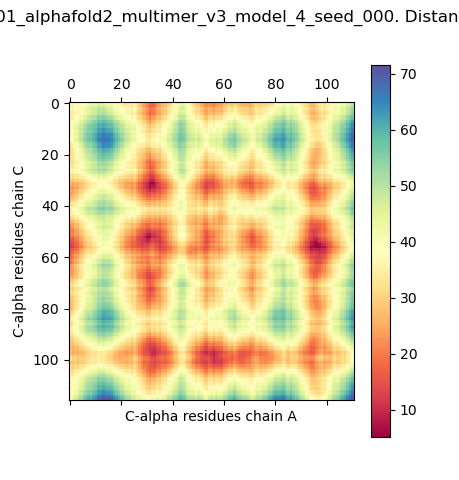


Figure: C-alpha distance plot of chain A vs chain C and chain B vs chain C

Comparing the plots from control to the predicted structure, we can observe that overall the plots are almost same, signifying that the assumptions made by us holds that the light and heavy chain folds properly after the SHM and that the relative configuration of the Pembrolizumab-PD1 complex remains the same.

* + - 1. *Improperly folded state after AlphaFold2 prediction*



Figure: AF predicted structure of state SYYYSDTWYNYNTNGYR aligned with pembrolizumab (control) state

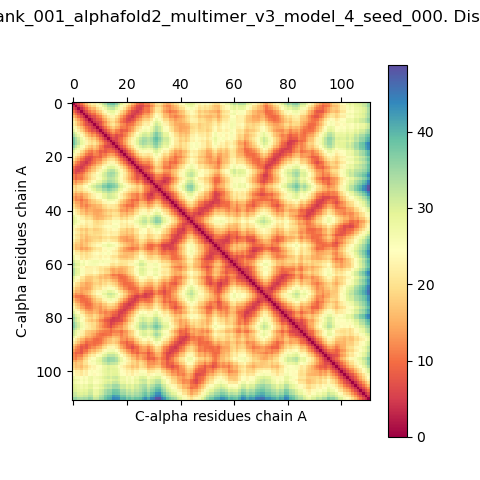
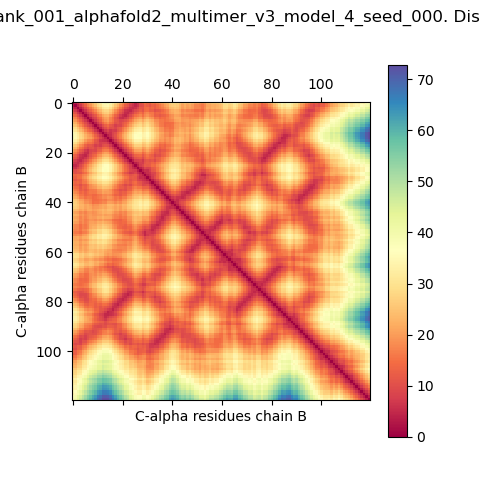


Figure: C-alpha distance plot of chain A vs chain A and chain B vs chain B

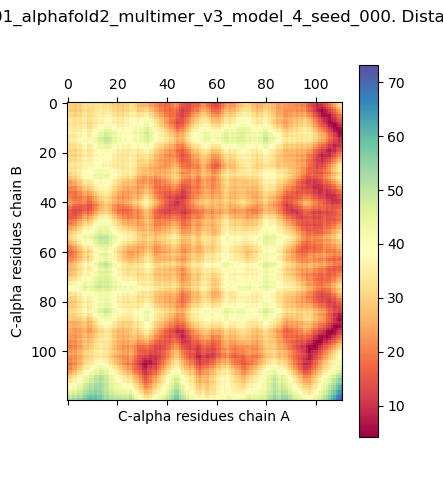


Figure: C-alpha distance plot of chain A vs chain B

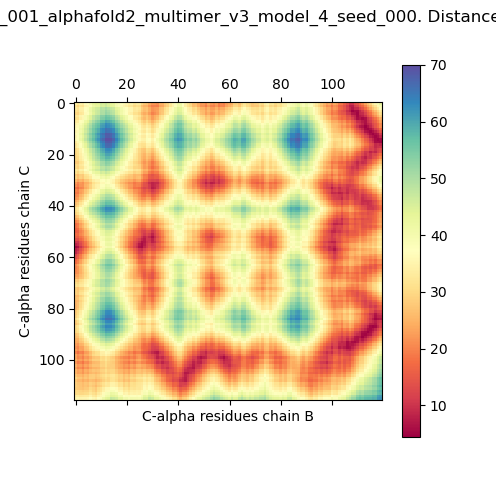
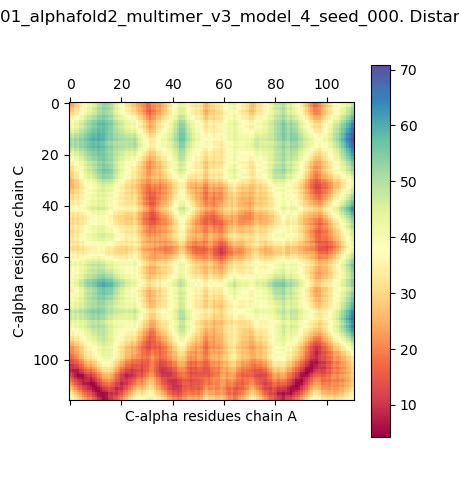


Figure: C-alpha distance plot of chain A vs chain C and chain B vs chain C

Our assumption doesn’t hold for all predicted structures, although for the majority of structures it does. We can observe that the light and heavy chains are improperly folded and the relative configuration of the mutated Pembrolizumab-PD1 complex has changed. Perhaps this can also be avoided if we perform predictions with multiple random seeds for our states.

We thought about implementing AF inside the main model rather than just for validation, but the AF is very slow in prediction and the ColabFold server for multiple sequence analysis (MSA) for which the local application sends request is limited to 10,000 requests per day and can be more restrictive, this can act as a big bottleneck for a faster simulation required for the restricted time provided.

1. **CONCLUSION**

We can observe that our Q-learning simulations were able to simulate the SHM, especially in case of mutations on heavy chain positions. This gives us the confidence to move forward with DQL and other deep reinforcement approaches, which may be better suited for our problem statement. We are currently working on a DQL model, but due to the complex nature of the problem it requires optimization to simulate SHM which was not possible in the time frame available.

Although the major objective of creating a RL model is successful for a restricted state space using Q-learning. It is important to search for alternative algorithms which may be better suited for the SHM process. The minor objective was accomplished as we were able to find multiple states with better binding affinity than Pembrolizumab according to assessment with PRODIGY, but these states need to be further validated and reduced using other tools to confidently provide with an antibody which can be engineered in a wet lab and tested experimentally to test its binding affinity relative to pembrolizumab.

The current model has a limitation that it requires a PDB with crystal structure of an antibody-antigen and the set of important interactions it makes with the antigen to implement it properly. This can be overcome if we can implement an algorithm to simulate V(D)J recombination before the SHM model.

There are also some tools available which can be used to create antibodies for an epitope (Raoufi E et al., 2020; Dunbar J et al., 2016), the SHM simulation can be performed after it to improve their binding affinity or can be used after wet lab pipeline of finding an antibody for the given epitope (Zhou J et al., 2024).

1. **REFERENCES**
2. Dunbar, J., Krawczyk, K., Leem, J., Marks, C., Nowak, J., Regep, C., Georges, G., Kelm, S., Popovic, B., & Deane, C. M. (2016). SAbPred: a structure-based antibody prediction server. *Nucleic Acids Research*, *44*(W1), W474–W478. https://doi.org/10.1093/NAR/GKW361
3. Horita, S., Nomura, Y., Sato, Y., Shimamura, T., Iwata, S., & Nomura, N. (2016). High-resolution crystal structure of the therapeutic antibody pembrolizumab bound to the human PD-1. *Scientific Reports 2016 6:1*, *6*(1), 1–8. https://doi.org/10.1038/srep35297
4. Kastritis, P. L., Moal, I. H., Hwang, H., Weng, Z., Bates, P. A., Bonvin, A. M. J. J., & Janin, J. (2011). A structure-based benchmark for protein–protein binding affinity. *Protein Science*, *20*(3), 482–491. https://doi.org/10.1002/PRO.580
5. Kastritis, P. L., Rodrigues, J. P. G. L. M., Folkers, G. E., Boelens, R., & Bonvin, A. M. J. J. (2014). Proteins Feel More Than They See: Fine-Tuning of Binding Affinity by Properties of the Non-Interacting Surface. *Journal of Molecular Biology*, *426*(14), 2632–2652. https://doi.org/10.1016/J.JMB.2014.04.017
6. Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization*.
7. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*.
8. Myung, Y., Pires, D. E. v, & Ascher, D. B. (2022). CSM-AB: graph-based antibody-antigen binding affinity prediction and docking scoring function. *Bioinformatics (Oxford, England)*, *38*(4), 1141–1143. https://doi.org/10.1093/BIOINFORMATICS/BTAB762
9. Odegard, V. H., & Schatz, D. G. (2006). Targeting of somatic hypermutation. *Nature Reviews Immunology 2006 6:8*, *6*(8), 573–583. https://doi.org/10.1038/nri1896
10. Papavasiliou, F. N., & Schatz, D. G. (2002). Somatic hypermutation of immunoglobulin genes: Merging mechanisms for genetic diversity. *Cell*, *109*(2 SUPPL. 1), S35–S44. https://doi.org/10.1016/S0092-8674(02)00706-7
11. Peterson, C., Denlinger, N., & Yang, Y. (2022). Recent Advances and Challenges in Cancer Immunotherapy. *Cancers*, *14*(16). https://doi.org/10.3390/CANCERS14163972
12. Raoufi, E., Hemmati, M., Eftekhari, S., Khaksaran, K., Mahmodi, Z., Farajollahi, M. M., & Mohsenzadegan, M. (2020). Epitope Prediction by Novel Immunoinformatics Approach: A State-of-the-art Review. *International Journal of Peptide Research and Therapeutics*, *26*(2), 1155. https://doi.org/10.1007/S10989-019-09918-Z
13. Raucci, R., Laine, E., & Carbone, A. (2018). Local Interaction Signal Analysis Predicts Protein-Protein Binding Affinity. *Structure (London, England : 1993)*, *26*(6), 905-915.e4. https://doi.org/10.1016/J.STR.2018.04.006
14. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction, 2nd ed. In *Reinforcement learning: An introduction, 2nd ed.* The MIT Press.
15. Vangone, A., & Bonvin, A. M. J. J. (2015). Contacts-based prediction of binding affinity in protein–protein complexes. *ELife*, *4*(JULY2015). https://doi.org/10.7554/ELIFE.07454
16. Vangone, A., & Bonvin, A. M. J. J. (2017). PRODIGY: A Contact-based Predictor of Binding Affinity in Protein-protein Complexes. *Bio-Protocol*, *7*(3). https://doi.org/10.21769/BIOPROTOC.2124
17. Xue, L. C., Rodrigues, J. P., Kastritis, P. L., Bonvin, A. M., & Vangone, A. (2016a). PRODIGY: a web server for predicting the binding affinity of protein–protein complexes. *Bioinformatics*, *32*(23), 3676–3678. https://doi.org/10.1093/BIOINFORMATICS/BTW514
18. Xue, L. C., Rodrigues, J. P., Kastritis, P. L., Bonvin, A. M., & Vangone, A. (2016b). PRODIGY: a web server for predicting the binding affinity of protein–protein complexes. *Bioinformatics*, *32*(23), 3676–3678. https://doi.org/10.1093/BIOINFORMATICS/BTW514
19. Yang, Y. X., Huang, J. Y., Wang, P., & Zhu, B. T. (2023). AREA-AFFINITY: A Web Server for Machine Learning-Based Prediction of Protein-Protein and Antibody-Protein Antigen Binding Affinities. *Journal of Chemical Information and Modeling*, *63*(11), 3230–3237. https://doi.org/10.1021/ACS.JCIM.2C01499/ASSET/IMAGES/LARGE/CI2C01499\_0005.JPEG
20. Yuan, Y., Chen, Q., Mao, J., Li, G., & Pan, X. (2023). DG-Affinity: predicting antigen–antibody affinity with language models from sequences. *BMC Bioinformatics*, *24*(1), 1–12. https://doi.org/10.1186/S12859-023-05562-Z/FIGURES/6
21. Zhou, J., Le, C. Q., Zhang, Y., & Wells, J. A. (2024). A general approach for selection of epitope-directed binders to proteins. *Proceedings of the National Academy of Sciences*, *121*(19). https://doi.org/10.1073/PNAS.2317307121