**Code applied in calculations for Manuscript**

The code in the repository is divided into folders according to their use-case. The folders contain code, input and output datasets as examples can be found in this google drive link: https://drive.google.com/drive/folders/1qDimifO\_S-1yp-pIfXydyw\_2vkxTWF6B?usp=sharing. The code should be executed according to the order indicated:   
**1. Folder ID: Tree\_Grouping:** Code used for phylogenetic analyses.   
**1.1. branchify.py:** Branchify takes a newick tree file and outputs grouped sequence IDs based on the provided distance threshold.   
 **Input:** Tree\_input\_sample.txt   
 **Output:** Branchify\_output\_sample.txt

**1.2. downsize\_branchified\_groups.py:** This program takes a partitioned group txt file (output of the above) and divides the large groups (larger than the cutoff size of 50, designated the “ceiling”) into multiple smaller groups (e.g., 50 sequences per group). The code also removes the groups that are smaller than the cutoff number.   
 **Input:** Branchify\_output\_sample.txt   
 **Output:** Downsize\_branchified\_groups\_output\_sample.txt

**2. Folder ID:AA\_seqs\_group\_attributes:** The code takes grouping information from **Tree\_Grouping** and a fasta file (aligned amino acids) and combines them into a .csv file where sequences are assigned their groups as determined above.

**2.1 fas\_to\_csv.py:** Takes a fasta file as input and outputs .csv file where each residue is assigned a cell.   
 **Input:** Example\_Fasta\_file.fasta   
 **Output:** Example\_output\_csv\_fas\_to\_csv.csv

**2.2 seq\_match\_branch.py:** Takes the output of fas\_to\_csv and the output of **downsize\_branchify.py** (Grouping information). It outputs a csv file where each sequence has a group identifier.   
 **Inputs:** Example\_output\_csv\_fas\_to\_csv.csv  
 Seq\_match\_branch\_input\_groups\_50s.txt  
 **Output:** Example\_output\_seq\_match\_branch.csv

**3. Folder ID:V\_metric:** Code used to obtain the Volatility metric *V*.

**3.1 stdev\_cal.py:** The code assigns each amino acid a value according to its hydropathy score. Each residue, including a PNGS, has a different value according to a modified Black and Mould score (built into the code). The standard deviation of the hydropathy scores across a cluster of 50 sequences is calculated.  
 **Input:** stdev\_cal\_input.csv  
 **Output:** sample\_stdev\_output.csv

**4. Folder ID: R\_metric:** Code used to obtain the volatility metric *R*.

**4.1 stdev\_cal.py:** Code calculates the standard deviation of the hydropathy scores across a cluster of 50 sequences.   
 **Input:** stdev\_cal\_input.csv  
 **Output:** sample\_stdev\_output.csv

**4.2 FisherExact.java:** The code calculates the co-mutability of any two spike positions. The code uses as input a matrix that contains all spike positions (in columns) and all 50-sequence clusters (in rows). The values describe the absence (0) or presence (1) of volatility in each cluster at each position. The output is a matrix that contains the P-values in the tests.  
 **Input:** test\_stdev\_fisher\_input.csv  
 **Output:** FisherExact\_output.csv

**4.3. fisherMatrix\_to\_Column.py:** Code used to transform the above matrix into a column format and to filter out position pairs based on their values (position pairs with P-values smaller than a defined threshold are listed)  
 **Input:** FisherExact\_output.csv  
 **Output:** R\_metric\_network\_output.csv

**4.4 R.py**: Code computes the *R* metric from a log-transformed volatility input.  
 **Inputs:** Sample\_R\_input.csv  
 fishermatrix\_to\_column\_output\_log\_transformed.csv  
 **Output:** sample\_stdev\_output.csv

**5. Folder ID: D\_metric:** Code used to obtain the volatility metric *D*.

**5.1. Subfolder: Trimer Distance:** The folder contains code used to calculate the minimal distance (in Å) between any two positions of spike.  
**5.1.1: trimer\_eucli\_distance.py:** Code to calculate the above Euclidean distances.   
 **Input:** input for trimer\_eucli\_dist.csv  
 **Output:** output\_trimer\_eucli\_dist.csv

**5.1.2: convert\_to\_pos\_matrix.py:** Code to obtain the shortest distance between any two positions and accounts for the three protomers of the protein.   
 **Input:** output\_trimer\_eucli\_dist.csv  
 **Output:** output\_convert\_to\_post\_matrix.csv

**5.1.3: fisherMatrix\_to\_Column.py:** Code used to transform the above matrix into a column format and to filter out position pairs with Euclidean distances above a user-defined threshold.  
 **Input:** output\_convert\_to\_post\_matrix.csv  
 **Output:** final\_6zgi\_output.csv

**5.2 stdev\_cal.py:** Code calculates the standard deviation of the hydropathy scores across a cluster of 50 sequences.  
 **Input:** stdev\_cal\_input.csv  
 **Output:** sample\_stdev\_output.csv

**5.3 D.py:** Code calculates the metric *D*, which describes the total volatility at all positions within a given distance, which is weighted by the reciprocal of the distance.  
 **Inputs:** input\_for\_D\_6zgi\_distances\_reciprocal.csv  
 Sample\_D\_input.csv  
 **Output:** sample\_D\_output.csv

**6. Folder ID: Lineage\_permutation\_MDS:** Code calculates the lineage specificity of n-feature vectors based on Euclidean distances between them and their lineage-association.  
**6.1 specificity.py**  
 **Inputs:** MDS\_sample\_input.csv  
 **Output**: P-values are indicated in the Console.

**7. Folder ID: V\_permutation:** Calculates the clustering of volatility on the spike trimer.

**7.1** **implement.py:** Invokes the permutation test from Permutation.py. The code applies values as inputs as well as a Euclidean distance matrix. The output is the P-value that indicates the presence of an environment that is more volatile than any randomly selected environment.

**Inputs:** 10.27.21 Baseline\_Vp.csv (Volatility Values)  
 10.25.21 No\_gaps.csv (Distance matrix)  
 10.25.21 missing data.csv (Positions with no distance information, gaps in trimer structure)  
 **Output:** List of Neighbors associated with the position of interest.  
 Null Statistic  
 P-value