AIMS

Release 0.6

Boughter

Jan 18, 2023

CONTENTS

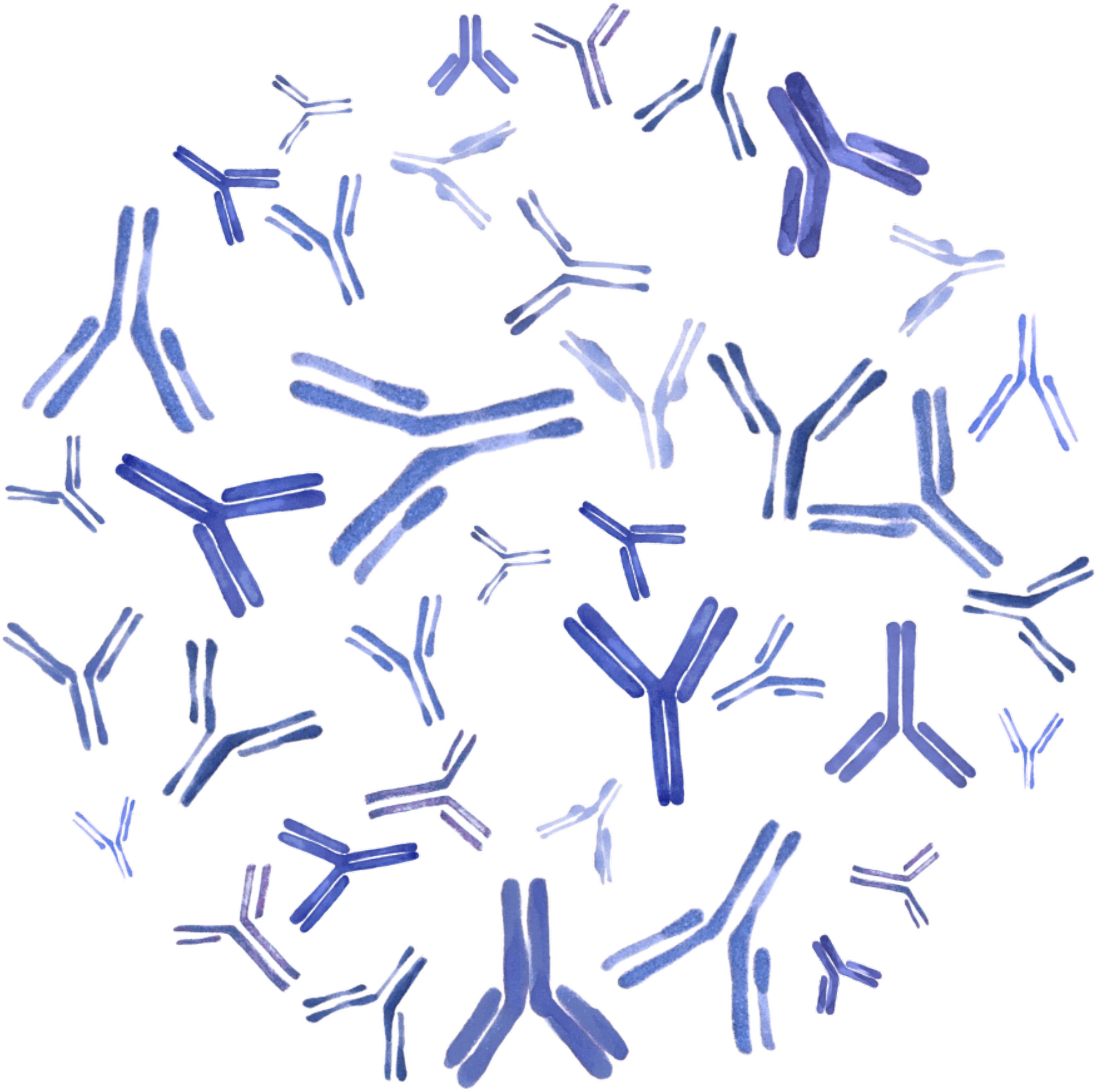


Fig. 1: AIMS Graphic created by Anna Borowska (see https://annazofiaborowska.com/)

Note: This readthedocs site is still under development. Hopefully what is included thus far helps users, but more information is coming soon!

AIMS, an Automated Immune Molecule Separator, was originally developed to identify differences between two distinct antibody repertoires, but has since expanded to become a multi-purpose repertoire analysis tool. Currently the AIMS analytical tools can be applied to immunoglobulin (Ig) molecules such as T cell receptors and antibodies, major histocompatibility complex (MHC) and MHC-like molecules, immunopeptidomic data, and broadly to any data presented in a multi-sequence alignment format. This documentation will teach users how to use AIMS to:

- **Get started** following the *Installation Instructions*.
- **Analyze repertoires with no programming experience required** in a user-friendly format with the *AIMS GUI*

Walkthrough.

- **Characterize the key features of immune repertoires** using the AIMS biophysical characterization tools through the *AIMS Basics* section. This biophysical characterization is the central feature of AIMS, taking properly formatted files (see *Input Formatting*) and applying analytical tools from the *Core Functionalities* of AIMS for downstream manipulation.
- **Identify biophysically distinct repertoire clusters** within single repertoires or comparing across multiple repertoires using the *AIMS Cluster* features.
- **Build off the AIMS analysis with your own custom features** by taking advantage of the *AIMS Jupyter Notebooks*.

AIMS is a python package distributed in both a notebook and GUI format. Those wishing to use the GUI, particularly those relatively new to programming, can follow the installation instructions. Example data is provided in AIMS/app/ab_testData and AIMS/app/mhc_testData, and an example of an application of AIMS can be seen in this peer-reviewed article: https://elifesciences.org/articles/61393

Note: When publishing analysis from this software, please cite:

Boughter CT, Borowska MT, Guthmiller JJ, Bendelac A, Wilson PC, Roux B, Adams EJ. Biochemical Patterns of Antibody Polyreactivity Revealed Through a Bioinformatics-Based Analysis of CDR Loops. eLife. 2020. DOI: 10.7554/eLife.61393

CHAPTER

ONE

CONTENTS

1.1 Installation Instructions

1.1.1 Initial Installation for Beginners

These instructions will help novice programmers install all necessary packages and programs needed to run a python program. Mac/Linux OS preferred, these steps outline installation via Anaconda. Other installations should be supported but have had limited testing.

Note: If you are a more advanced programmer, skip to the *Installation of AIMS & Required Packages* section

1. Install Anaconda (https://www.anaconda.com/products/individual) to manage the python packages we're going to be using. This can be a fairly large package, so if space is at a premium for your computer, you can instead install miniconda (https://docs.conda.io/en/latest/miniconda.html). NOTE: Windows users should likely install the full Anaconda package, for a contained environment to run python programs from.
2. Test that your conda install is working properly by creating a conda environment. Windows OS users, you will likely do this within the Anaconda application. Mac/Linux users, open the terminal application. Once terminal is open, type:

```
conda create -n aims-env python=3.7
```

If anaconda/miniconda is installed properly, a Y/N prompt should appear. Type "y" then hit the "enter key" and you will create a conda environment.

3. If you haven't already, you should download the code from https://github.com/ctboughter/AIMS.
4. Next, navigate to the new folder created from this repository. First, open up terminal and enter the environment created earlier by typing:

```
conda activate aims-env
```

Note: Windows users will have to activate the environment from within Anaconda

You should now see a little extra bit of text on your terminal command line that looks something like "(aims-env)". If this didn't work for some reason, an error message should pop up, otherwise assume you're fine.

Use terminal to navigate into the newly downloaded folder. If you've never used terminal before, you can type in "cd" and then drag and drop the folder into the terminal. Doing so should automatically populate the "path" to the folder. Then hit enter.

When I do this, my terminal line reads:

Hopefully you see something similar (replacing my user name with your own), and when you hit enter you can move to the next step.

1.1.2 Installation of AIMS & Required Packages

The above section focused on the installation of the necessary software to run python programming, as well as basic navigational instructions. In this section, instructions for installing requisite packages and running the GUI Are discussed.

1. From within the AIMS master directory, type in the terminal:

```
./app/install_packages.sh
```

Warning: If you want to use Python 3.9, do not run the above command or install packages in the code block below. Instead, see closed issue #1 on the GitHub page [https://github.com/ctboughter/AIMS/issues/1] for working package versions.

This bash script should run after typing in this command, and you'll be prompted with a bunch of [y]/n prompts, for which you should consistently enter "y" then the "enter key".

Note: For more advanced users, you shouldn't need to use these EXACT package versions, save for Biophython and SciKit-Learn. However, using these versions should guarantee proper functionality of AIMS

If the install_packages.sh script doesn't work, and you get some kind of an error instead of the prompts, type each of these lines (or copy/paste) one by one, hitting enter after each one:

```
conda install -c conda-forge biopython=1.76
conda install -c conda-forge scipy=1.4.1
conda install pandas=1.0.3
conda install numpy=1.18.1
conda install matplotlib=3.1.3
conda install scikit-learn=0.22.1
conda install seaborn=0.10.1
conda install -c conda-forge kivy=2.0.0
```

2. Everything should now be installed, you should now be able to open up the software! Navigate to the app in terminal by typing:

```
cd app
```

3. Launch the GUI with:

```
python aims.py
```

From there, the GUI should open. A step by step instruction guide for GUI usage can be found in the *AIMS GUI Walkthrough* section. If you don't want to be bothered reading instructions, the app should prevent most major errors. If a "next" button is grayed out, make sure you've pressed all of the analysis buttons on the bottom of the current AIMS app screen.

If you're a more advanced user and would prefer a more customizable experience, check out the *AIMS Jupyter Notebooks* section.

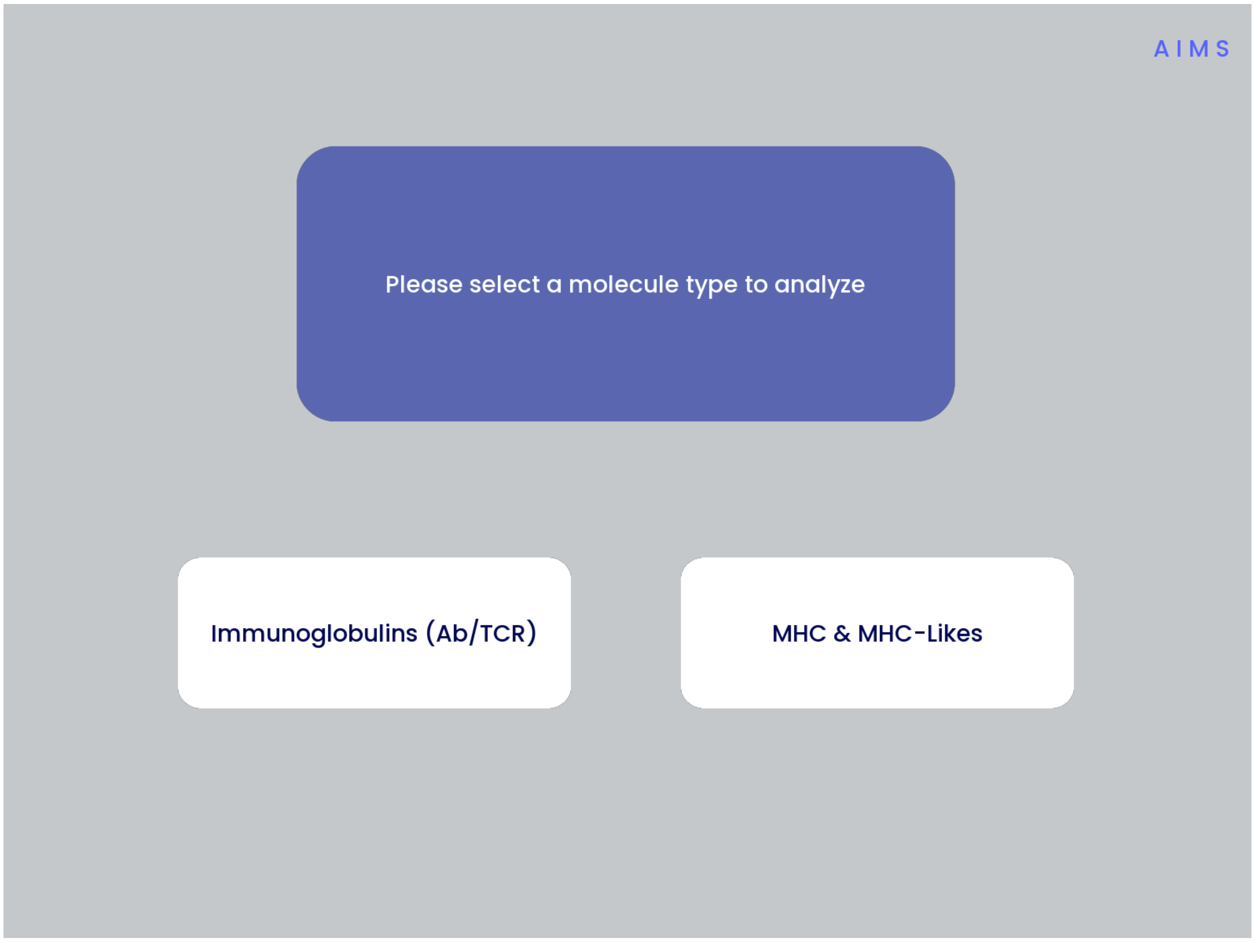
Lastly, if you're generally interested in an overview of what AIMS does and how it works, refer to the *AIMS Basics*.

1.2 AIMS GUI Walkthrough

The AIMS analysis pipeline has been used to analyze a wide range of molecular species, including TCRs, antibodies, MHC molecules, MHC-like molecules, MHC-presented peptides, viral protein alignments, and evolutionarily conserved neuronal proteins. The GUI is currently only capable of analyzing these first four molecular species, with more analysis options hopefully available in the future.

This section will provide a step-by-step walkthrough outlining how to use the AIMS Graphical User Interface (GUI) through screenshots of the app. The focus will be primarily on how to interface with the GUI, how files are saved when using the GUI, and tips and tricks for a smooth AIMS experience. Before starting the GUI, you may want to check out the *AIMS Basics* and review the *Input Formatting* and *Core Functionalities*.

When launching the GUI, this screen should be the first thing that you see:



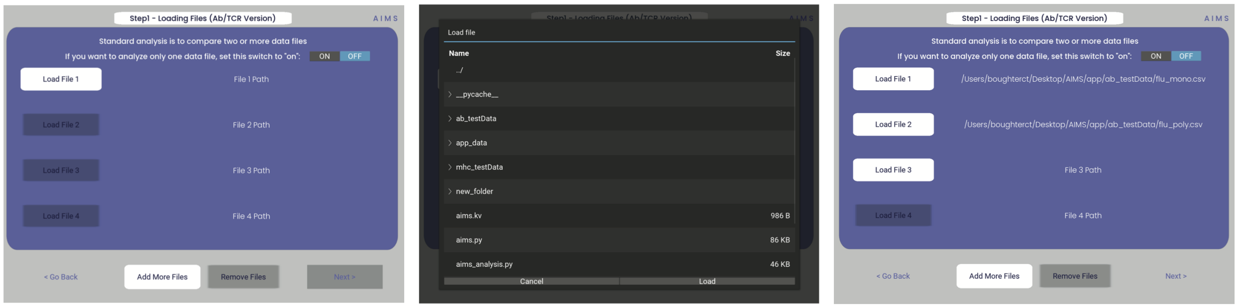
The navigation through the AIMS GUI is linear and button-based, allowing users to simply click through all of the analysis. If you'd like to analyze antibody (Ab) or T cell receptor (TCR) sequences, start with the *Immunoglobulin Analysis with AIMS* section. If you'd like to analyze MHC and MHC-like molecules, skip down to the *MHC and MHC-Like Analysis with AIMS* section.

1.2.1 Immunoglobulin Analysis with AIMS

This section is specifically for the analysis of T cell receptors and antibodies. The analysis and formatting are identical for each of these receptor types, and one could even analyze these receptor types simultaneously, if for some reason one wanted to.

Step 1: Loading in Data

Assuming the input data is already properly formatted, everything should easily flow through the GUI and generate all data required. The first screen after selecting the Ig analysis should look like the first panel in the below image:



Step1 - Loading Files (Ab/TCR Version)

Standard analysis is to compare two or more data files

If you want to analyze only one data file, set this switch to "on":

File 1 Path

File 2 Path

File 3 Path

File 4 Path

Load file

Name	Size
../	
__pycache__	
ab_testData	
app_data	
mhc_testData	
new_folder	
aims.kv	986 B
aims.py	86 KB
aims_analysis.py	46 KB

/Users/boughterct/Desktop/AIMS/app/ab_testData/flu_mono.csv

/Users/boughterct/Desktop/AIMS/app/ab_testData/flu_poly.csv

Once you click one of those light gray "Load File" buttons, you should get a screen that looks like the image in the second panel. From this, you can click through your directories and select the file you want to load in. If the file has been properly loaded, the File Path should be updated to reflect the file location.

Note: If following along with this walkthrough, select the "ab_testData" directory and load in flu_mono.csv and flu_poly.csv

If you would like to analyze only one dataset using AIMS, change the "ON/OFF" switch in the top right-hand corner to "ON". Otherwise, the AIMS GUI will require at least two datasets to be loaded into the analysis. Additionally, if the data of interest is spread across more than four files, click the "add more files" option to increase the number of file slots on the screen. As of AIMS_0.5.5, a max of 9 files may be analyzed at once.

Step 2: Define Names and Outputs

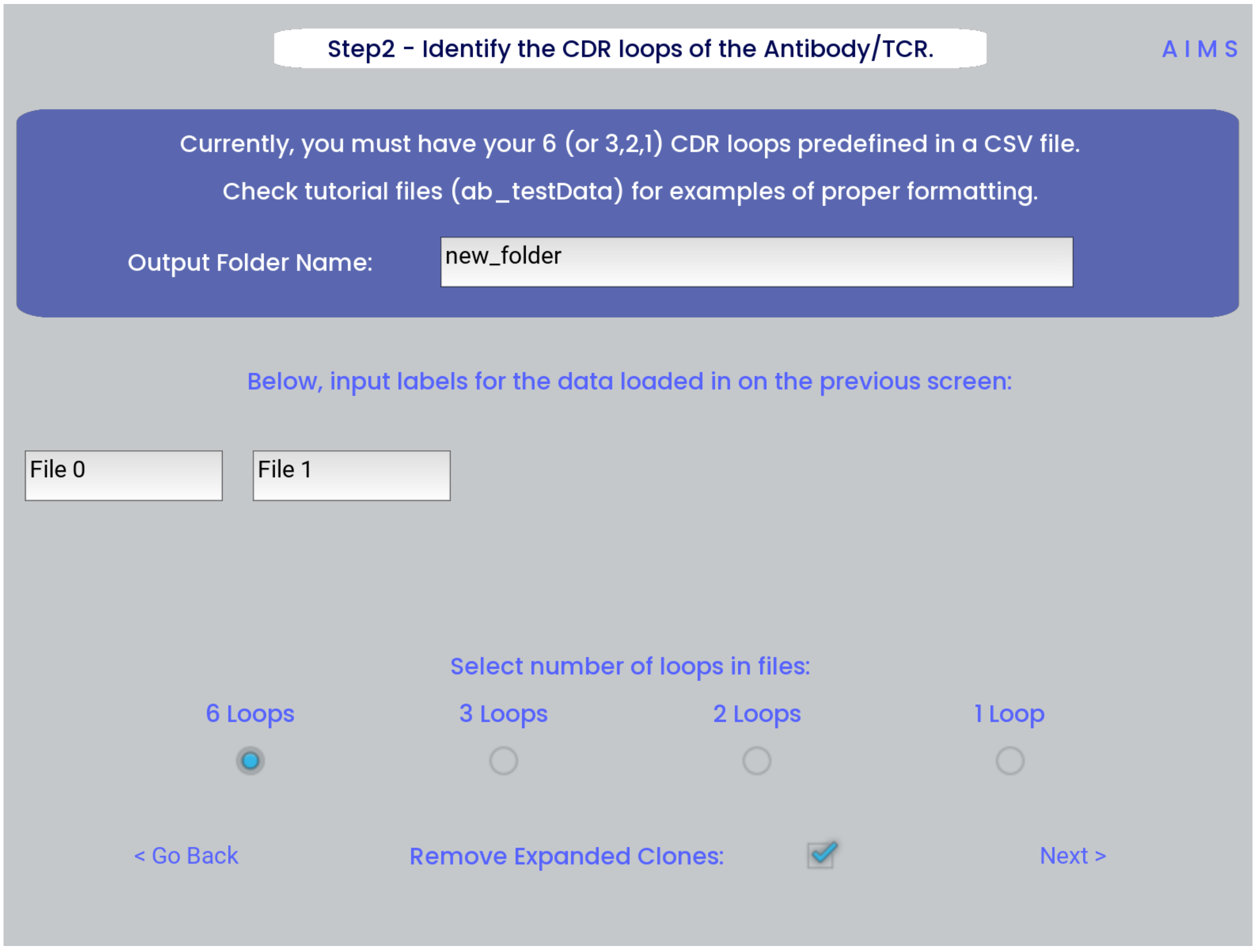
In this step, we define the folder which the outputs are saved to, the labels that will accompany the datasets in figure legends, and the number of CDR loops in the input data. Further, the AIMS GUI removes expanded clones and sequence duplicates by default. Uncheck the "Remove Expanded Clones" box if you'd like to include degenerate receptors.

Warning: Defining the output directory is very important, as by default files are overwritten if the AIMS analysis is run multiple times. Ideally each new run of AIMS should be output to a new directory, with a descriptive title.

Step 3: Generate the Sequence Matrix

In this step, we begin the dive in to the automated portion of the "Automated Immune Molecule Separator", generating an AIMS-encoded matrix form of the sequences in our dataset. Most steps from here on require a simple clicking of a button, here "Get Matrix", and moving on to the next step. Users are given the option to change the AIMS aligment scheme, with "Center Align" as the default option. See the *Core Functionalities* section of the documentation for more information on these alignment schemes. As an example, both the bulge alignment (center panel), and the central alignment (right panel) are highlighted below.

Congrats! You've generated your first piece of data using this software. You might notice that your image quality is poor for figures shown in the app, this is because the software shows *png files. Don't worry, both a *png and a higher-



Step2 - Identify the CDR loops of the Antibody/TCR.

Currently, you must have your 6 (or 3,2,1) CDR loops predefined in a CSV file.

Check tutorial files (ab_testData) for examples of proper formatting.

Output Folder Name: new_folder

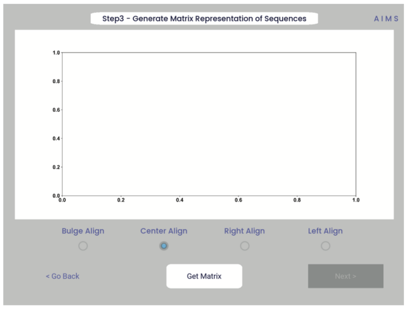
Below, input labels for the data loaded in on the previous screen:

File 0 File 1

Select number of loops in files:

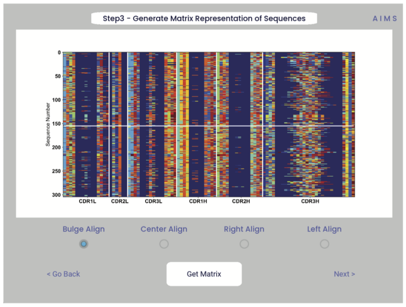
● 6 Loops ○ 3 Loops ○ 2 Loops ○ 1 Loop

Remove Expanded Clones: ☑



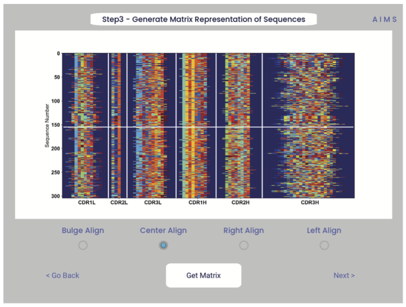
Step3 - Generate Matrix Representation of Sequences

○ Bulge Align ● Center Align ○ Right Align ○ Left Align



Step3 - Generate Matrix Representation of Sequences

● Bulge Align ○ Center Align ○ Right Align ○ Left Align



Step3 - Generate Matrix Representation of Sequences

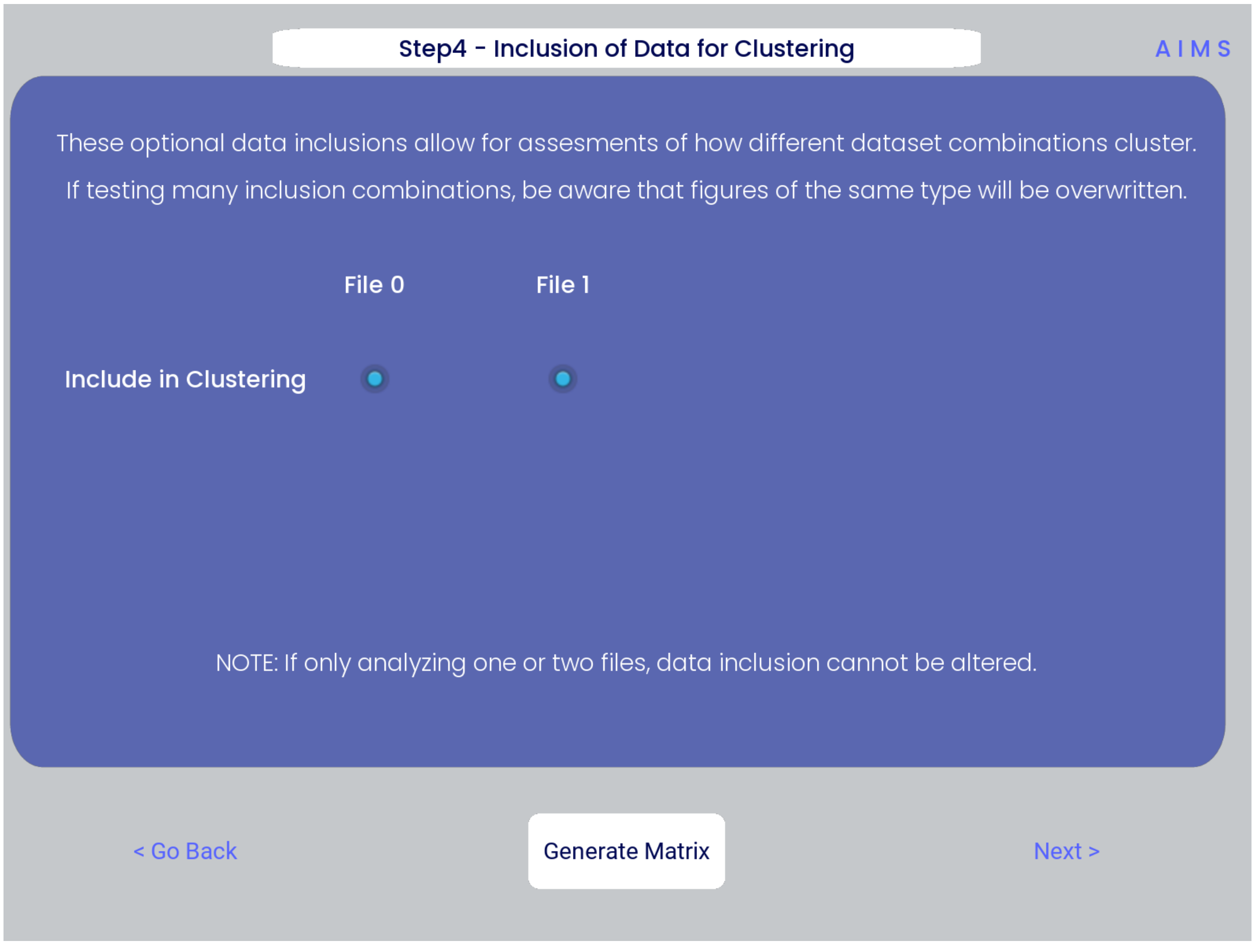
○ Bulge Align ● Center Align ○ Right Align ○ Left Align quality *pdf version of the plot are saved in whichever directory you specified in Step 2. This is true for subsequently generated figures.

Additionally, as you move through the sequential steps of the GUI, keep in mind that **all generated figures have corresponding raw data saved to a *.dat file**, should the user want to re-plot the data using a different color scheme, different plotting application, etc. For screen 3, the output figures are matrix.png and matrix.pdf, while the ouput raw data is saved as raw_matrix.dat.

Note: Whichever alignment is chosen at this step will be used for all downstream analysis from this point. In other words, analysis of the central alignment may be different from analysis of the left or right aligments.

Step 4: Generate High-Dimensional Biophysical Matrix

In this step, we generate the high-dimensional biophysical property matrix that will be used in all downstream analysis. We then have the option to include or exclude files from the clustering that will happen in the next step. If only one or two datasets are included in the analysis, all input data must be included in the clustering. Again, we simply press the "Generate Matrix" button, shown below, and then users can move on to the next step.



Step4 - Inclusion of Data for Clustering

These optional data inclusions allow for assesments of how different dataset combinations cluster.

If testing many inclusion combinations, be aware that figures of the same type will be overwritten.

	File 0	File 1
Include in Clustering	●	●

NOTE: If only analyzing one or two files, data inclusion cannot be altered.

Note: Don't worry if this step takes a little while, especially for larger datasets. This will be the slowest and most memory-intensive step in the analysis.

While most users may not want to leave any datasets out of the clustering of Step 5, there are some interesting applications of AIMS where the exclusion of certain datasets has interesting impacts on receptor clustering. It is important to remember that the UMAP and PCA dimensionality reduction steps are strongly dependent on the input data. Learn more about this input data dependence in the *AIMS Cluster* section.

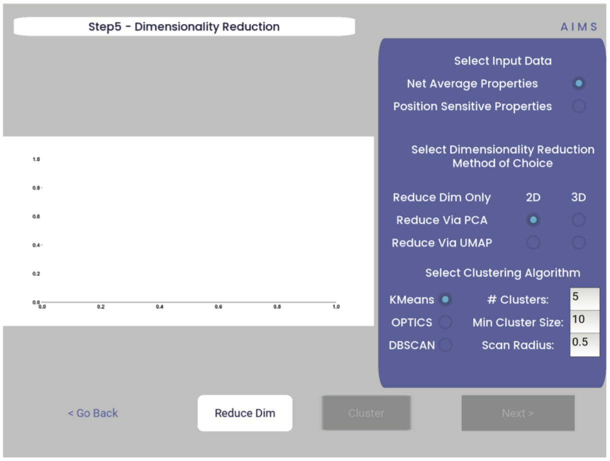
Step 5: Dimensionality Reduction and Receptor Clustering

The goal in this step is to take that large biophysical property matrix generated in the previous step, and reduce this high-dimensional matrix down to two or three composite dimensions, and then cluster the receptors projected onto this space based upon distance in this projected space. This step is perhaps the most involved in the GUI, with the most customizable options. Hence the dedicated *AIMS Cluster* section detailing the possibilities for this analysis.

First, the user must decide if they would like to reduce dimensionality on Net Average Properties, i.e. the biophysical properties averaged across entire receptors, or on the Position Sensitive Properties, i.e. the amino acid biophysical properties at every position on every receptor.

Next, the algorithm used for this dimensionality reduction must be chosen. Users can choose either Principal Component Analysis (PCA) or Uniform Manifold Approximation and Projection (UMAP), and additionally choose to visualize these projections in two- or three-dimensions. Once these options are chosen, click the "Reduce Dim" button to visualize these options. More options can be tested and the projection re-visualized as many times as the user desires.

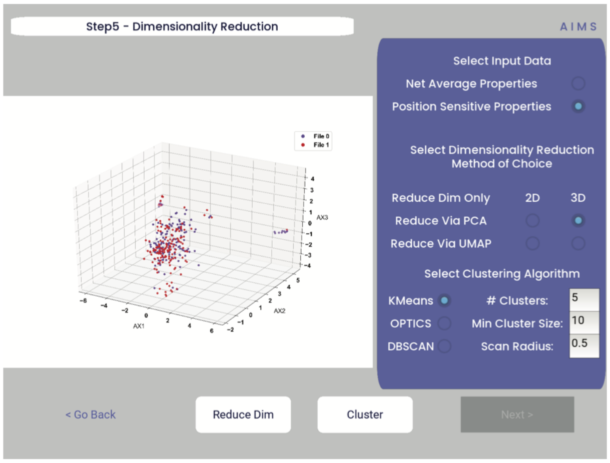
Lastly, the data is then clustered using one of three algorithms, either K-Means, OPTICS, or DBSCAN clustering. Users must also define, for each of these algorithms, a tunable parameter that determines the size of the clusters generated. We can see each of these options, and the default values for the tunable parameters, in the screenshots below.


Step5 - Dimensionality Reduction

Select Input Data
Net Average Properties ●
Position Sensitive Properties ○

Select Dimensionality Reduction Method of Choice
Reduce Dim Only 2D 3D
Reduce Via PCA ● ○
Reduce Via UMAP ○ ○

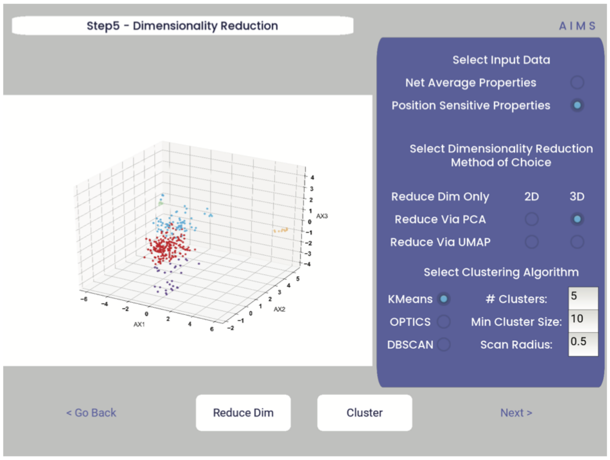
Select Clustering Algorithm
KMeans ● # Clusters: 5
OPTICS ○ Min Cluster Size: 10
DBSCAN ○ Scan Radius: 0.5


Step5 - Dimensionality Reduction

Select Input Data
Net Average Properties ○
Position Sensitive Properties ●

Select Dimensionality Reduction Method of Choice
Reduce Dim Only 2D 3D
Reduce Via PCA ○ ●
Reduce Via UMAP ○ ○

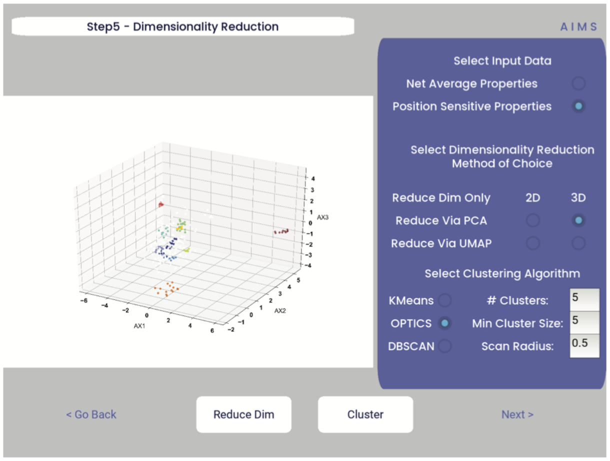
Select Clustering Algorithm
KMeans ● # Clusters: 5
OPTICS ○ Min Cluster Size: 10
DBSCAN ○ Scan Radius: 0.5


Step5 - Dimensionality Reduction

Select Input Data
Net Average Properties ○
Position Sensitive Properties ●

Select Dimensionality Reduction Method of Choice
Reduce Dim Only 2D 3D
Reduce Via PCA ○ ●
Reduce Via UMAP ○ ○

Select Clustering Algorithm
KMeans ● # Clusters: 5
OPTICS ○ Min Cluster Size: 10
DBSCAN ○ Scan Radius: 0.5


Step5 - Dimensionality Reduction

Select Input Data
Net Average Properties ○
Position Sensitive Properties ●

Select Dimensionality Reduction Method of Choice
Reduce Dim Only 2D 3D
Reduce Via PCA ○ ●
Reduce Via UMAP ○ ○

Select Clustering Algorithm
KMeans ○ # Clusters: 5
OPTICS ● Min Cluster Size: 5
DBSCAN ○ Scan Radius: 0.5

For more detail on how these dimensionality reduction and clustering algorithms work, as well as details on the tunable parameters, please see the *AIMS Cluster* section.

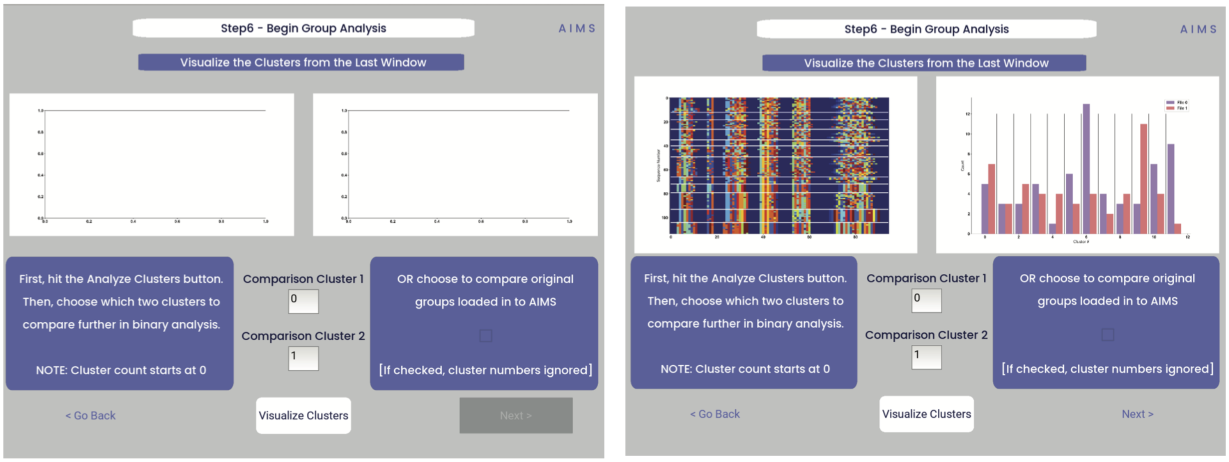
In the above screenshots, we see first the default screen (top left), then the three-dimensional PCA projection (top right), followed by a Kmeans clustering with 5 clusters (bottom left), and lastly an OPTICS clustering with a minimum cluster size of 5 (bottom right). Users should note that Kmeans will cluster all sequences in the dataset, while OPTICS and DBSCAN will exlude sequences that are not found at a sufficient density in the projection. These unclustered sequences are grayed out in the resultant displayed figure.

There is no one right answer to determining the "best" dimensionality reduction or clustering algorithm, so users are encouraged to try a range of options to determine which combination makes the most sense for their data. Importantly for this step, generated figures and data from each dimensionality reduction and clustering algorithm are **not overwritten**. You will see in the output directory descriptive filenames that correspond to each option used. Importantly however, for each given clustering algorithm only one file will be saved. If, for instance, you cluster your data using the Kmeans algorithm with "# Clusters" set to 5, then run it again with "# Clusters" set to 10, the output figures and data will reflect only the "# Clusters = 10" configuration. Lastly, raw data outputs "umap_3d.dat" or "pca_3d.dat" give the location of each sequence in the reduced dimensionality projection. Raw data outputs for the clustering steps "optics_3d.dat" or "dbscan_3d.dat" reflect the cluster membership of each given sequence, with the order of sequences preserved based upon your input datasets.

Note: Whichever projection and clustering algorithm the user is currently viewing when moving to the next step is what will be used in all downstream analysis

Step 6: Visualize and Analyze Clustered Sequences

At this stage, we visualize the clustered sequences from the previous step. First, the user must hit the "Visualize Cluster" button, then after visual inspection of the clusters and sequences, the "comparison clusters" can be selected. The goal of this step is to determine whether the user wants to compare two biophysically distinct clusters which were identified in the previous step, or compare across the input datasets. We can see in the screenshot below how this works:


Step6 - Begin Group Analysis
AIMS
Visualize the Clusters from the Last Window
< Go Back
Visualize Clusters
Next >

First, hit the Analyze Clusters button. Then, choose which two clusters to compare further in binary analysis.

NOTE: Cluster count starts at 0

Comparison Cluster 1: 0

Comparison Cluster 2: 1

OR choose to compare original groups loaded in to AIMS

□

[If checked, cluster numbers ignored]

After the cluster visualization is complete, we see in the right panel, left figure that the matrix from step 3 is rearranged to reflect the clustered sequences, with higher sequence conservation (colors in the matrix) evident within each cluster. In the right figure, we see the sequence count of each input dataset in each cluster.

From this information, the user can determine which clusters they would like to analyze by entering values in the "Comparison Cluster" boxes. The cluster count starts at zero, and the user can infer the last cluster number from the figure on the right. The amino acid sequences from the selected clusters will be saved to the output directory as "clust2seq_#.txt" where the "#" is the cluster number for each respective dataset.

If the user instead is still most interested in comparing the input datasets, the checkbox on the right side of the screen can be checked, ignoring the clustering of the data (but still saving the results in the output directory!).

Warning: The clustering algorithms are stochastic, and so cluster ID and cluster membership may change each time the software is run. For instance, in this walkthrough I use clusters 10 and 11 for downstream analysis, but users trying to replicate this analysis may have different sequences in clusters 10 and 11. This is important both for comparisons in this walkthrough as well as creating reproducible analysis.

Step 7: Define Comparison Classes

Note: This screen is skipped when cluster analysis is chosen, rather than original group analysis

Here, we separate our loaded data into separate classes for downstream analysis, assuming the user opted not to compare clustered sequences. As a default, each loaded dataset is assigned to its own unique group, but the user may group these datasets however they choose by assigning matching group numbers to datasets they want analyzed together. For the immmunoglobulin analysis, the cluster comparison option is chosen, so this screen is not shown. To see the comparison class definition screen, jump to Step 7 in the *MHC and MHC-Like Analysis with AIMS*.

Warning: If comparing more than two distinct groups, some of the analysis will be unavailble. These analyses include mutual information analysis, amino acid frequency characterization, and linear discriminant analysis. Each of these analyses require binary classifications of the data.

Step 8: Visualize Averaged Position Sensitive Biophysical Properties

In this step we look at average biophysical properties as a function of sequence space, part of our special "positional encoding". At this stage in the walkthrough we won't bother showing the "before" snapshots of the GUI, as the only options are to press the button which generates the plot, and then move on to the next step. However, if you're trying to compare the results to the data we get in this walkthrough, the generated plots are quite useful:

Note: Standard deviations are not shown, and ideally these would be calculated via bootstrapping

The figure in this step is saved as "pos_prop.pdf/png", while the raw data is saved as "position_sensitive_mat#.dat" where the "#" again corresponds to the selected cluster number, or if comparing the original input datasets, the user-defined group number. This data file has as many rows as the number of sequences in the selected cluster or group, and has 61 x # AIMS positions columns. Ideally this data would be saved as a tensor of shape # sequences x 61 x # AIMS positions, however, this would require saving the data as a numpy object, which would be less friendly to other programming languages for replotting and reformatting. The 61 here refers to the 61 biophysical properties of AIMS (listed in the *Biophysical Properties*).

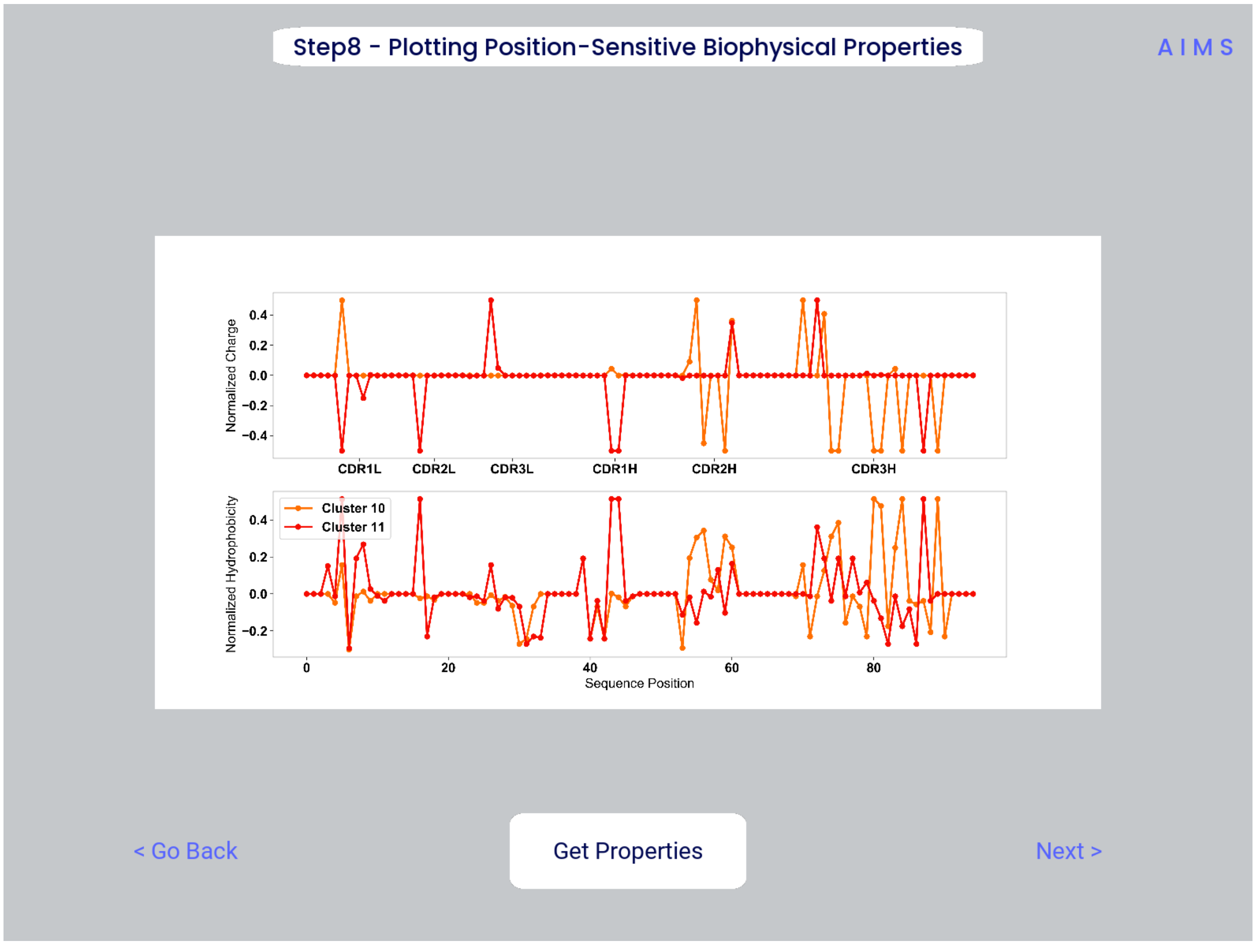
As a concrete example, cluster 10 in this walkthrough has 10 sequences, and 95 AIMS positions. So the first row of the position_sensitive_mat10.dat file here corresponds to the first sequence (which can be found in the "clust2seq_10.dat" output). The first 95 columns in this row correspond to the position sensitive charge (biophysical property 1 of 61). The next 95 columns correspond to the position sensitive hydrophobicity (biophysical property 2 of 61). And so on.

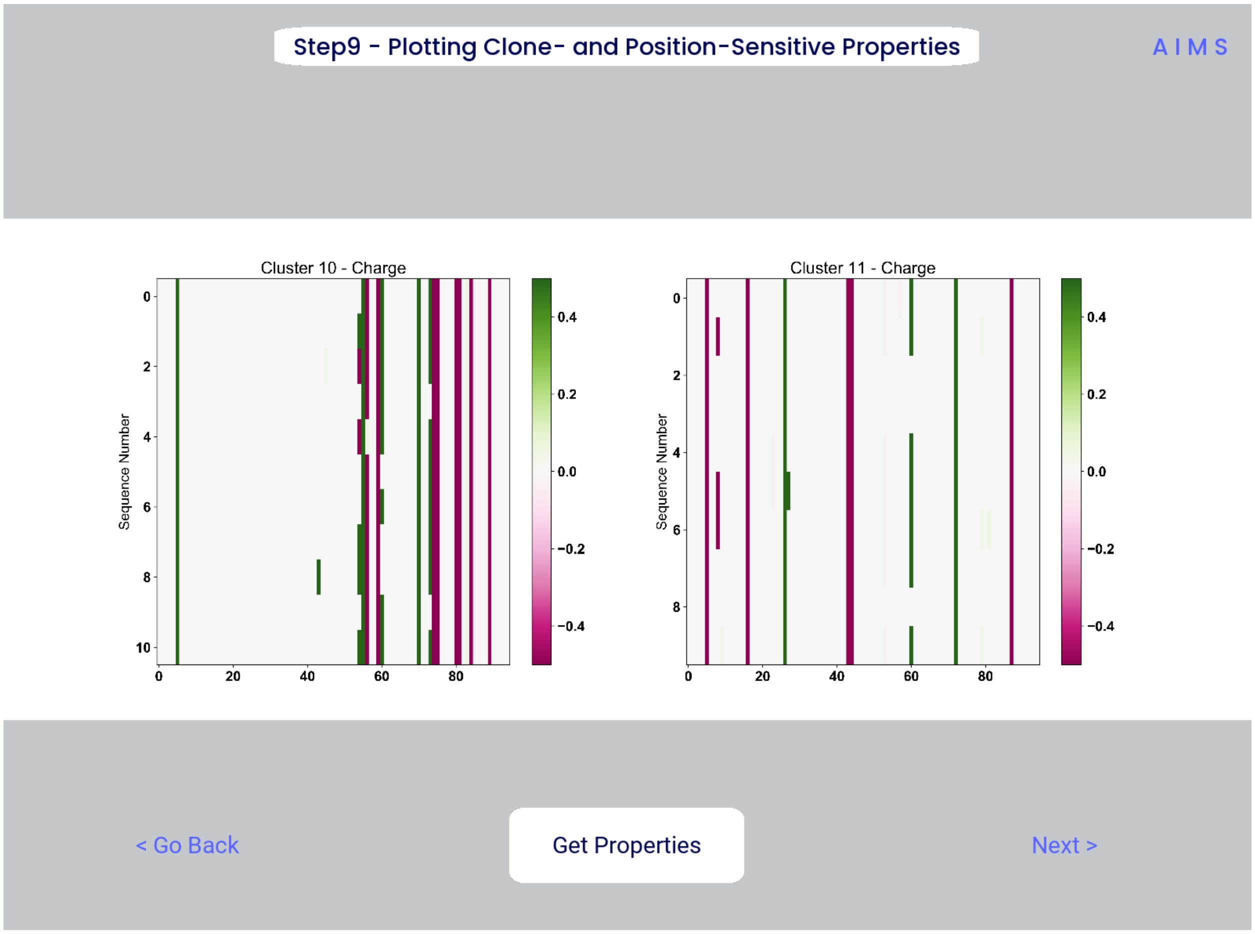
Step 9: Visualize Raw Position Sensitive Biophysical Properties

In this step, we visualize the position sensitive charge for all clones, not averaged. This figure can help provide a sense of how reliable the averages of the previous step are. Like all biophysical properties in AIMS, the charge is normalized, hence the minimum and maximum on the scales not equaling 1.

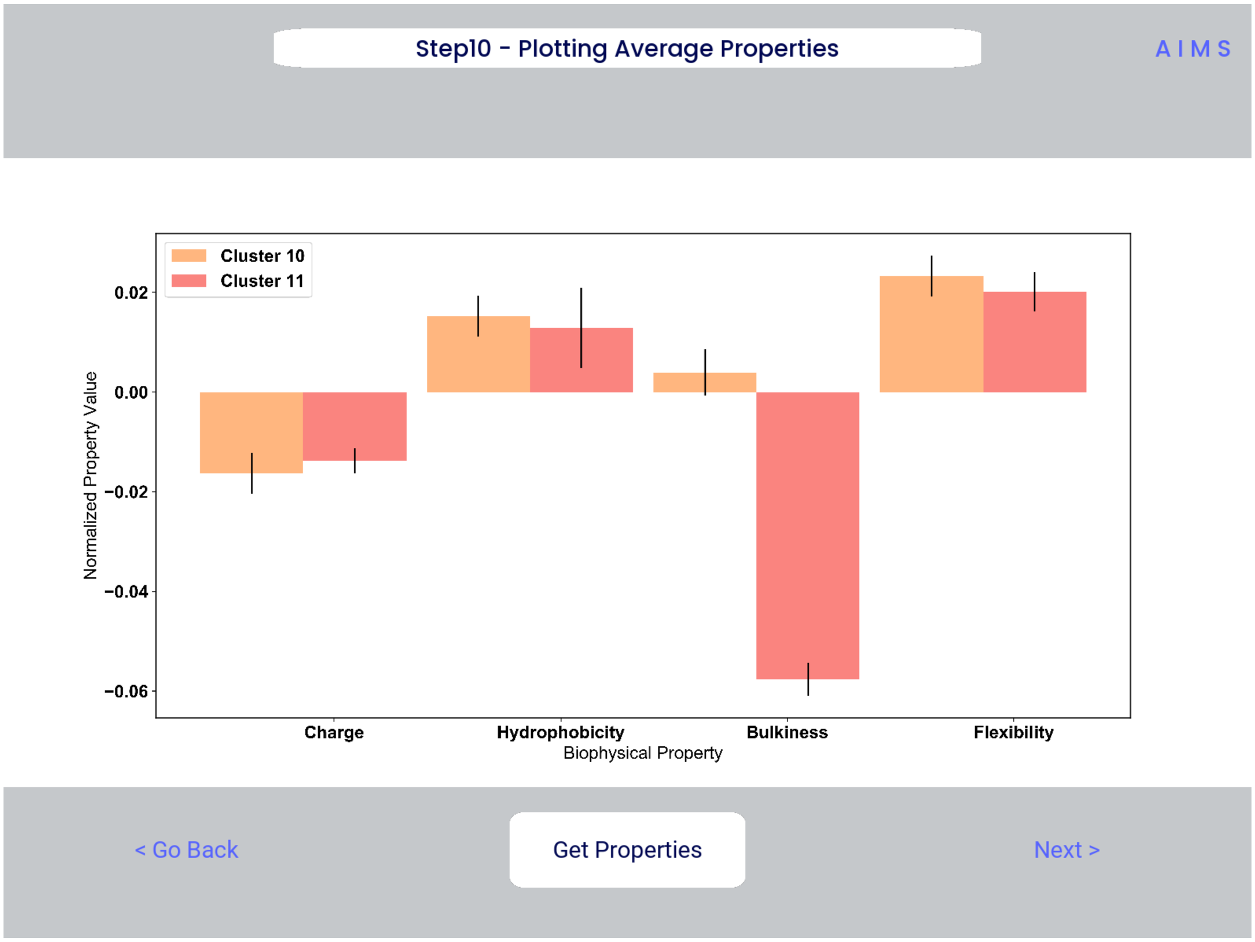
These figures are saved as "clone_pos_prop.pdf/png". This figure helps to understand the figure generated in Step 8, which is simply this figure averaged over the y-axis. It is additionally important to note that the positional encoding in Steps 8 and 9 are consistent with the alignment scheme selected in Step 3, either central, bulge, right, or left aligned.

Step 10: Visualize Net Biophysical Properties


Step8 - Plotting Position-Sensitive Biophysical Properties
A I M S
Normalized Charge
0.4
0.2
0.0
−0.2
−0.4
CDR1L
CDR2L
CDR3L
CDR1H
CDR2H
CDR3H
Normalized Hydrophobicity
Cluster 10
Cluster 11
0.4
0.2
0.0
−0.2
0
20
40
60
80
Sequence Position
< Go Back
Get Properties
Next >


Step9 - Plotting Clone- and Position-Sensitive Properties
AIMS
Cluster 10 - Charge
Sequence Number
0
2
4
6
8
10
0
20
40
60
80
0.4
0.2
0.0
−0.2
−0.4
Cluster 11 - Charge
Sequence Number
0
2
4
6
8
0
20
40
60
80
0.4
0.2
0.0
−0.2
−0.4
< Go Back
Get Properties
Next >

In this step, we are averaging the biophysical properties over all positions and all receptors. In other words, effectively averaging the figures generated in Step 9 over both the x- and the y-axes.



Note: A large standard deviation in these plots are to be expected, especially if users are analyzing original input datasets rather than selected cluster subsets

This figure is saved as “avg_props.pdf/png”, while statistical significance, as calculated using Welch’s t-Test with the number of degrees of freedom set to 1, is saved as “avg_prop_stats.csv”.

Step 11: Calculate Shannon Entropy

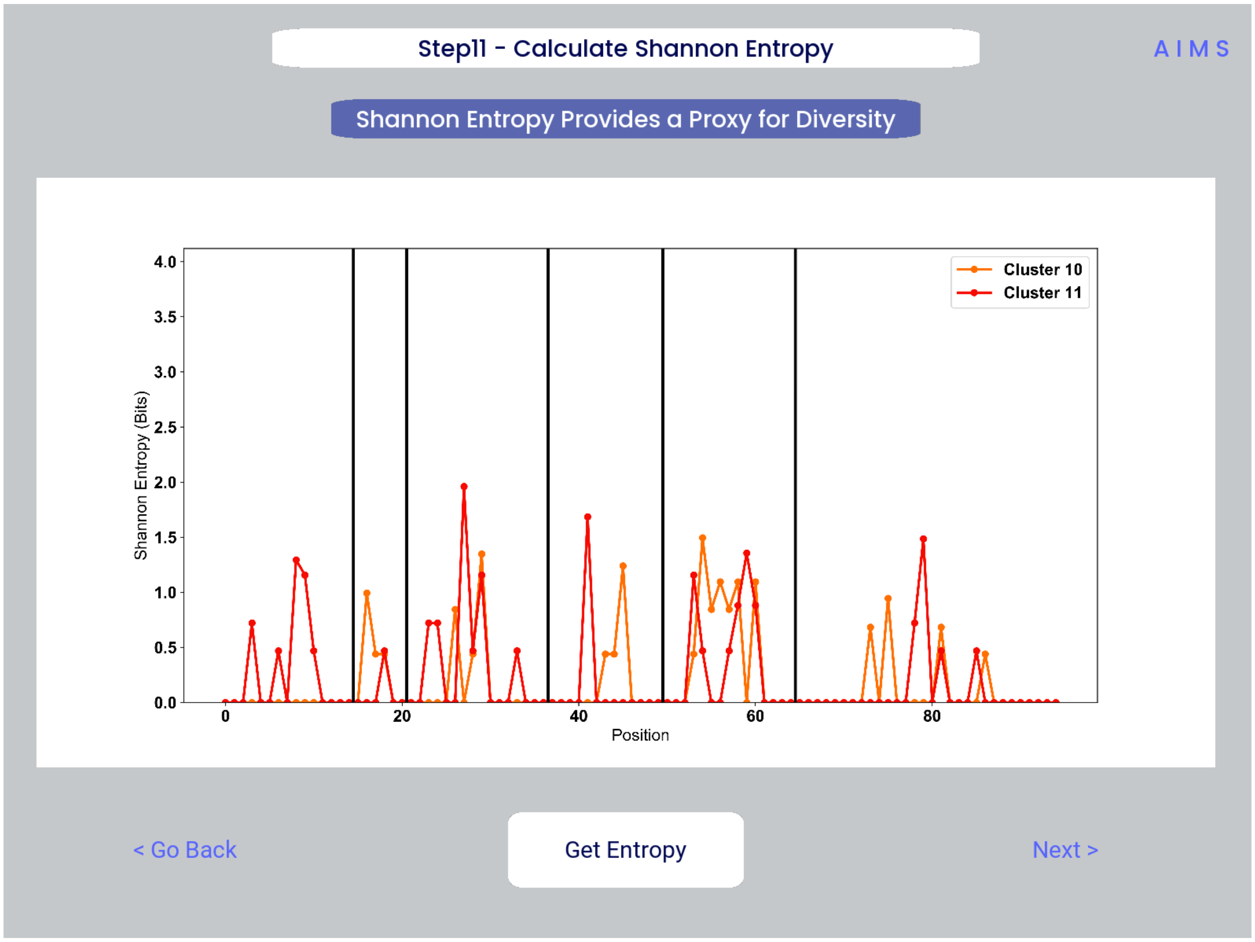
In this step, we are calculating the Shannon Entropy of the chosen datasets, effectively the diversity of the receptors as a function of position. For more information on the Shannon Entropy, as well as the Mutual Information discussed in the next step, view the Information Theory section of the *Core Functionalities*.

This figure is saved as “shannon.pdf/png”.

Note: Due to the requirement for a binary comparison in subsequent steps, this is the last GUI screen if users are comparing more than 2 groups

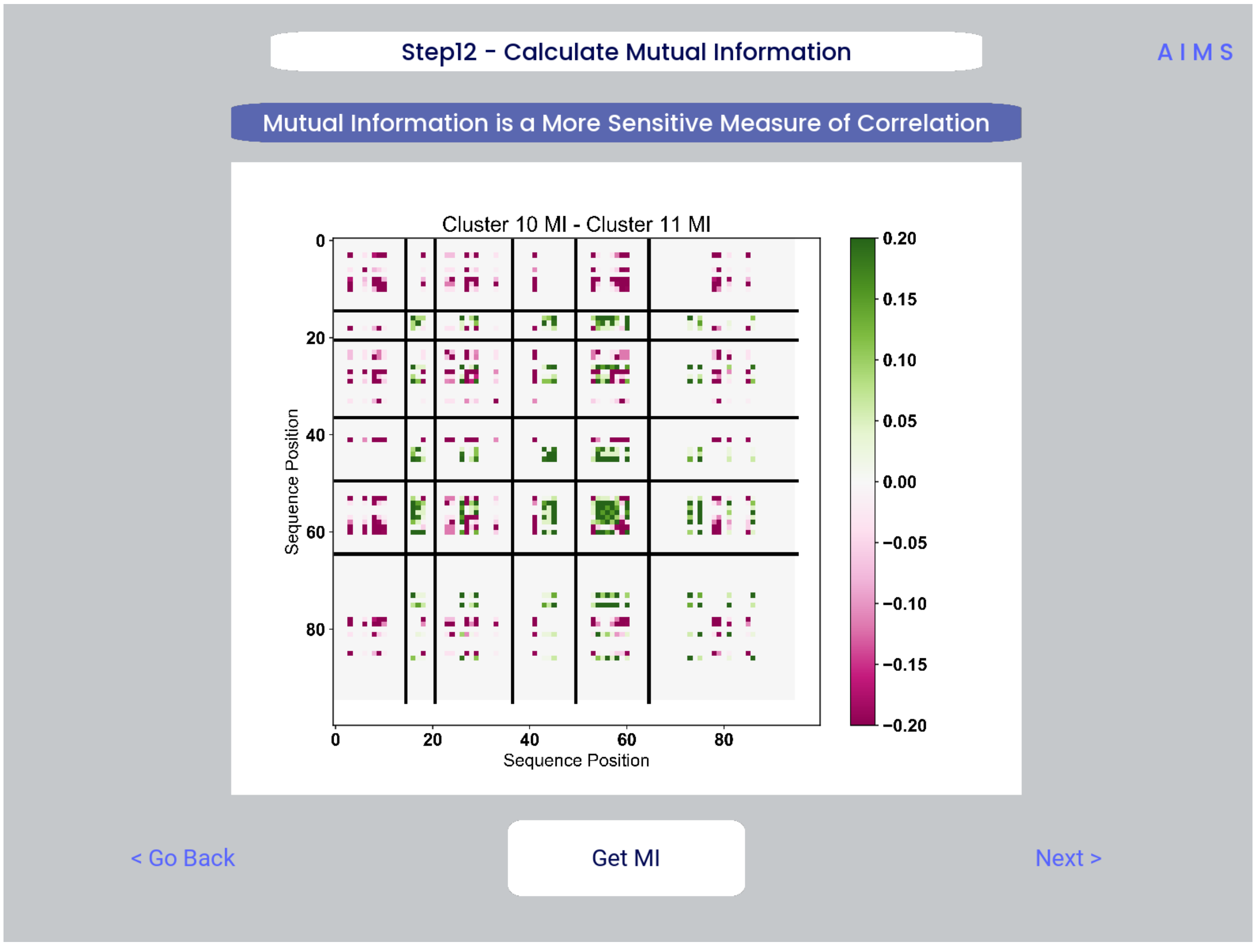
Step 12: Calculate Receptor Mutual Information

In this step, we calculate the mutual information between the individual posiitons in the AIMS matrix. The y-axis



Step11 - Calculate Shannon Entropy
A I M S
Shannon Entropy Provides a Proxy for Diversity
Cluster 10
Cluster 11
Shannon Entropy (Bits)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
Position
< Go Back
Get Entropy
Next >

provides the "given" amino acid, and the x-axis provides the amount of information we gain at every other position when we know the amino acid identity at the "given" position. We present this data as a difference between the mutual information of group 1 and the mutual information of group 2. The y-axis is measured in "Bits" the fundamental unit of information, with a positive value (green) indicating higher mutual information in the first group (here "Cluster 10") and a negative value (pink) indicating higher mutual information in the second group (here "Cluster 11").


Step12 - Calculate Mutual Information
A I M S
Mutual Information is a More Sensitive Measure of Correlation
Cluster 10 MI - Cluster 11 MI
Sequence Position
0
20
40
60
80
Sequence Position
0
20
40
60
80
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
< Go Back
Get MI
Next >

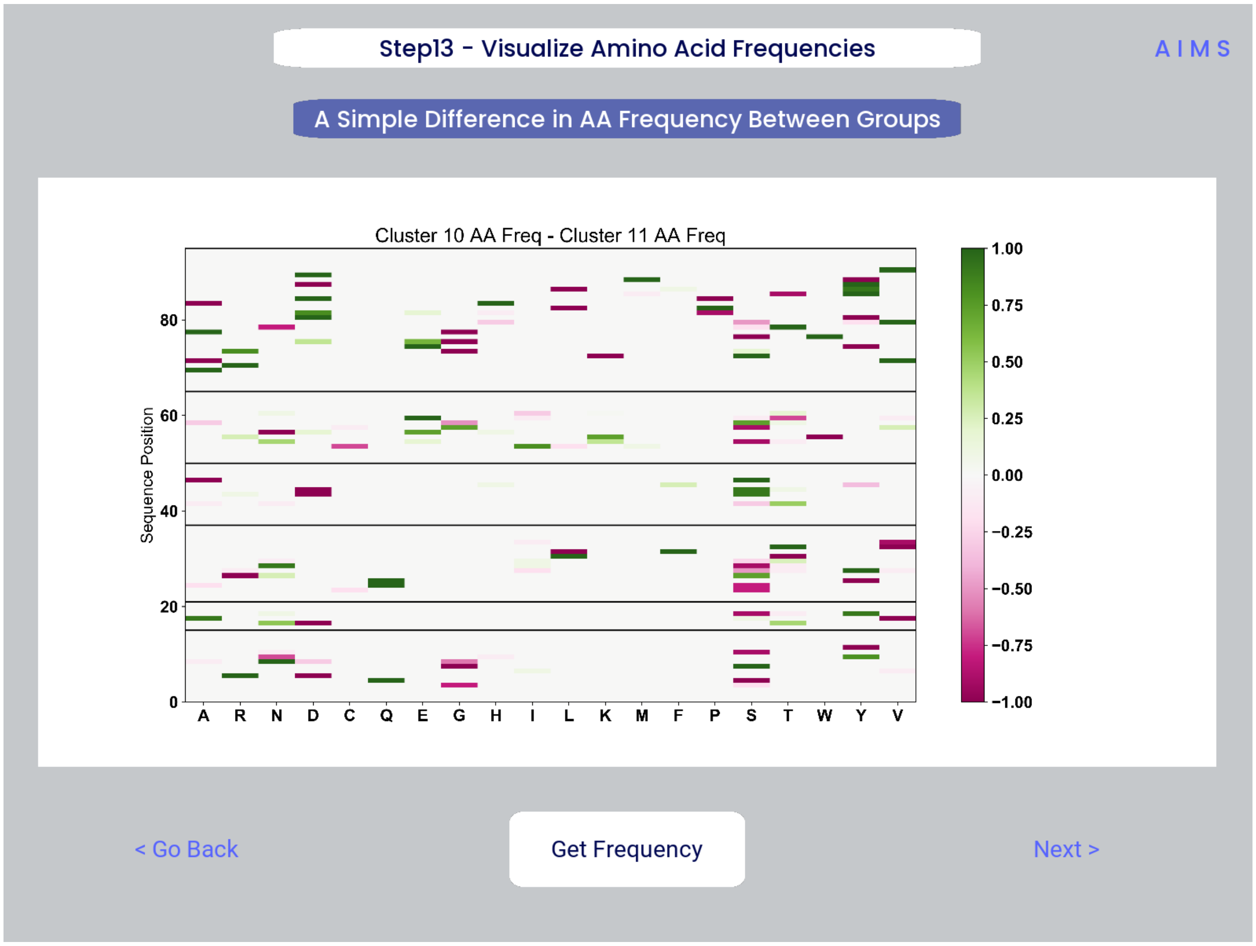
This figure is saved as "MI.pdf/png". The raw information matrices are saved as "MI_mat1.dat" amd "MI_mat2.dat", and should be symmetric matrices with shape # AIMS postions x # AIMS positions.

Note: Shannon entropy and mutual information are always positive, i.e. there is no "negative information", so we can be confident that negative values in this figure mean "higher mutual information in the second group" rather than "negative mutual information in the first group".

Step 13: Visualize Amino Acid Frequencies

In this step, we calculate the position sensitive amino acid frequnecy for each analyzed cluster or group, and plot the difference. The simply reports these differences in frequency, with a positive value (green) indicating higher frequency of a given residue at a given position in the first group (here "Cluster 10") and a negative value (pink) indicating higher frequency in the second group (here "Cluster 11").

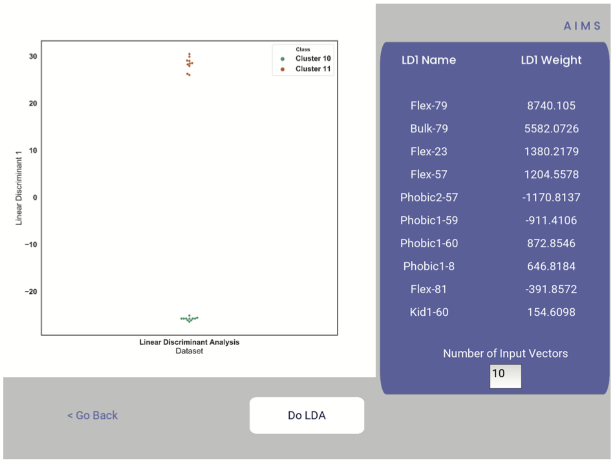
This figure is saved as "frequency.pdf/png". The raw position senesitive frequencies for each cluster or group are saved as "frequency_mat1.dat" amd "frequency_mat2.dat", with each row corresponding to the AIMS position, and each column corresponding to the amino acids in the same order as they are presented in the figure.



Step13 - Visualize Amino Acid Frequencies
A I M S
A Simple Difference in AA Frequency Between Groups
Cluster 10 AA Freq - Cluster 11 AA Freq
Sequence Position
0
20
40
60
80
A
R
N
D
C
Q
E
G
H
I
L
K
M
F
P
S
T
W
Y
V
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
< Go Back
Get Frequency
Next >

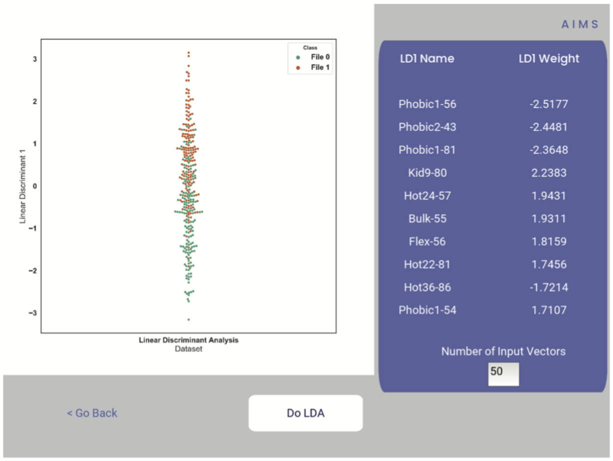
Step 14: Linear Discriminant Analysis

In the original eLife manuscript, linear discriminant analysis was used to classify antibody sequences as "polyreactive" or "non-polyreactive" (see https://elifesciences.org/articles/61393). In this step, we use the same framework to instead classify either the selected clusters or the user-defined groups analyzed in the previous steps. For a deeper description of linear discriminant analysis, see *Core Functionalities*. So, while in the eLife manuscript the linear discriminant is a proxy for polyreactivity, in the AIMS GUI the linear discriminant is a metric of "more like group 1" or "more like group 2". An example of overfit data (from a cluster analysis, left) and of a proper application of linear discriminant analysis (from a group analysis, right) can be seen below:



LD1 Name	LD1 Weight
Flex-79	8740.105
Bulk-79	5582.0726
Flex-23	1380.2179
Flex-57	1204.5578
Phobic2-57	-1170.8137
Phobic1-59	-911.4106
Phobic1-60	872.8546
Phobic1-8	646.8184
Flex-81	-391.8572
Kid1-60	154.6098

Number of Input Vectors: 10



LD1 Name	LD1 Weight
Phobic1-56	-2.5177
Phobic2-43	-2.4481
Phobic1-81	-2.3648
Kid9-80	2.2383
Hot24-57	1.9431
Bulk-55	1.9311
Flex-56	1.8159
Hot22-81	1.7456
Hot36-86	-1.7214
Phobic1-54	1.7107

Number of Input Vectors: 50

> **Warning:** Care must be taken not to overfit. If the number of input vctors is greater than (or similar to) the size of one of your datasets, you will likely overfit the data

The LD1 "names" and "weights" refer to the top ten weights that most strongly split the data. In other words, LDA not only functions as a classifier, it also works as a means to identify the biophysical features that best discriminate between two datasets. The generated figure is saved simply as "lda.pdf/png" while the raw data to recreate the plot is saved as "lda_data.dat". Lastly, the linear weights from which the linear discriminant is generated are saved as "lda_weights.dat". The AIMS GUI will show at most the top ten weights, but users can split their data using as many features as they choose (assuming this number is less than the available features).

> **Note:** You can tell that the left panel is overfit in part by the exaggerated weights, compared to the non-overfit weights in the right panel

END Ig Analysis

Congratulations for making it through the GUI walkthrough, and thanks again for using the software! Be sure to reach out if any part of this walkthrough is unclear or if there are questions/features you would like addressed in greater detail.

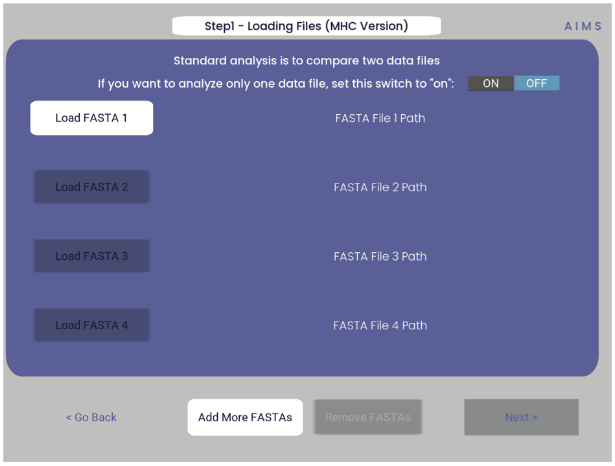
1.2.2 MHC and MHC-Like Analysis with AIMS

While a niche application of the software, AIMS readily extends to the analysis of any evolutionarily conserved molecules with specific regions of variability. MHC and MHC-like molecules fit very well into this category, and in the first published usage of AIMS, these moleclules were analyzed using the same tools as the immunoglobulin analysis. This section highlights the unique portions of the MHC analysis, and reiterates many of the points discussed in the above section for users only interested in the MHC and MHC-like analysis.

Note: Much of this documentation will be a verbatim repeat of the steps outlined above in the *Immunoglobulin Analysis with AIMS*, save for the first two steps which differ significantly

Step 1: Loading in Data

FASTA files should be aligned sequences, with a minimum of 2 sequences per file, and a minimum of 2 FASTA files per program run. For the MHCs, formatting should just be in normal FASTA format. For following along with the analysis, load in "mhc_testData/"cd1_seqs.fasta".


Step1 - Loading Files (MHC Version)

Standard analysis is to compare two data files

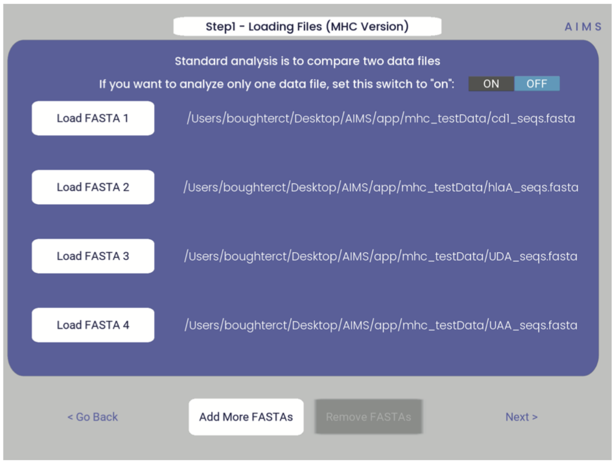
If you want to analyze only one data file, set this switch to "on": ON OFF

FASTA File 1 Path

FASTA File 2 Path

FASTA File 3 Path

FASTA File 4 Path


Step1 - Loading Files (MHC Version)

Standard analysis is to compare two data files

If you want to analyze only one data file, set this switch to "on": ON OFF

Load FASTA 1 /Users/boughterct/Desktop/AIMS/app/mhc_testData/cd1_seqs.fasta

Load FASTA 2 /Users/boughterct/Desktop/AIMS/app/mhc_testData/hlaA_seqs.fasta

Load FASTA 3 /Users/boughterct/Desktop/AIMS/app/mhc_testData/UDA_seqs.fasta

Load FASTA 4 /Users/boughterct/Desktop/AIMS/app/mhc_testData/UAA_seqs.fasta

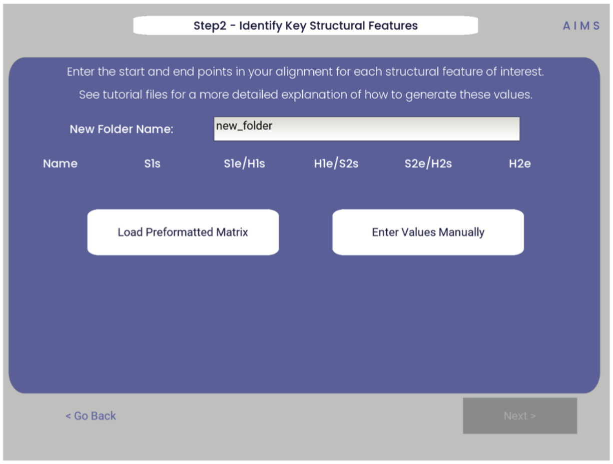
Step 2: Locate Helices and Strands

So this is my least favorite part of the software, but it turns out this is the most efficient way to do things. Here, we explicitly say where in the alignments the strands/helices start. In an attempt to make this slightly less annoying, I've made it possible to create pre-formatted matrices for repeated analysis.

For this example, from mhc_testData load in ex_cd1d_hla_uda_uaa_ji.csv. So for FASTA1, Strand 1 starts (S1s) at position 124, Strand 1 ends (S1e) at pos 167, Helix 1 starts (H1s) at this same position. And so on… Lastly, "new_folder" is where output figures will be saved. Change this to whatever you want your folder name to be. Each run overwrites the figures, so maybe change to "run1", "run2", etc.

How do we locate helices and strands? NOTE, for this tutorial, this step has been done already We first align molecules of interest within a single group We then take a representative molecule (here human CD1d) and put it through our favorite structure prediction (Phyre, PsiPred, etc.) When then go back and find where in the alignments a structural feature roughly begins Here S1 starts at "FPL" which occurs at alignment position 127. We add 3 amino acids of buffer space (optional, you can change this if you want) and you can see on the previous slide S1s = 124

Already figured out locations of Helices/Strands (based on provided FASTA files): For the ji_cartFish we have: 2,49,93,152,193 For the cd1d_seqs.fasta we have: 124,167,209,262,303 For the hlaA_seqs.fasta we have: 170,218,260,306,348 For cd1_ufa_genes.fasta: 22,66,105,158,199 For UAA or UDA fasta: 2,49,93,152,193 In the future, I hope to identify these helices and strands automatically within the software, but I haven't found anything suitable yet for doing so

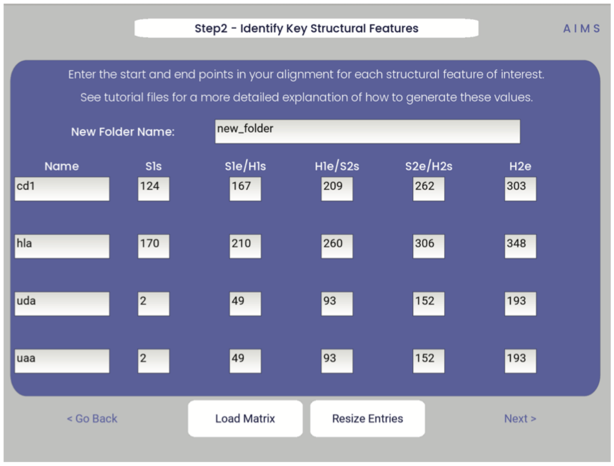

Step2 - Identify Key Structural Features

Enter the start and end points in your alignment for each structural feature of interest.

See tutorial files for a more detailed explanation of how to generate these values.

New Folder Name: new_folder

Name	S1s	S1e/H1s	H1e/S2s	S2e/H2s	H2e


Step2 - Identify Key Structural Features

Enter the start and end points in your alignment for each structural feature of interest.

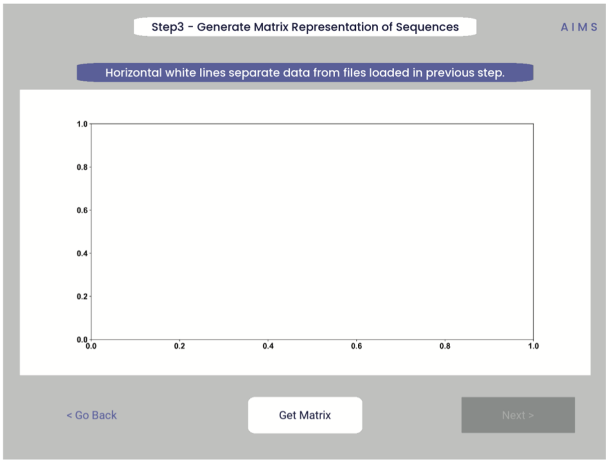
See tutorial files for a more detailed explanation of how to generate these values.

New Folder Name: new_folder

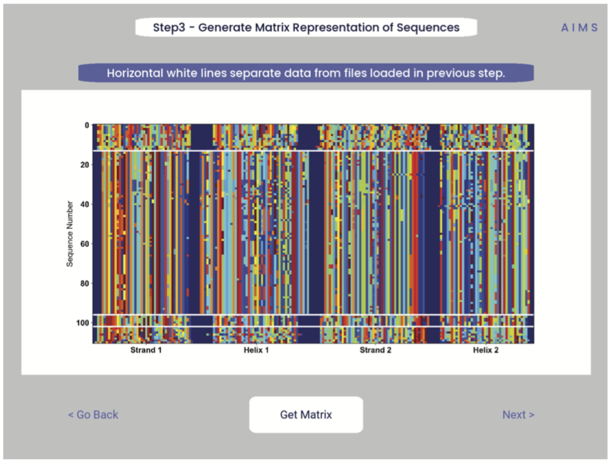
Name	S1s	S1e/H1s	H1e/S2s	S2e/H2s	H2e
cd1	124	167	209	262	303
hla	170	210	260	306	348
uda	2	49	93	152	193
uaa	2	49	93	152	193

Step 3: Generate the Sequence Matrix

In this step, we begin the dive in to the automated portion of the "Automated Immune Molecule Separator", generating an AIMS-encoded matrix form of the sequences in our dataset. Most steps from here on require a simple clicking of a button, here "Get Matrix", and moving on to the next step. Users are given the option to change the AIMS aligment scheme, with "Center Align" as the default option. See the *Core Functionalities* section of the documentation for more information on these alignment schemes. As an example, both the central alignment (center panel) and the bulge alignment (right panel) are highlighted below.


Step3 - Generate Matrix Representation of Sequences

Horizontal white lines separate data from files loaded in previous step.


Step3 - Generate Matrix Representation of Sequences

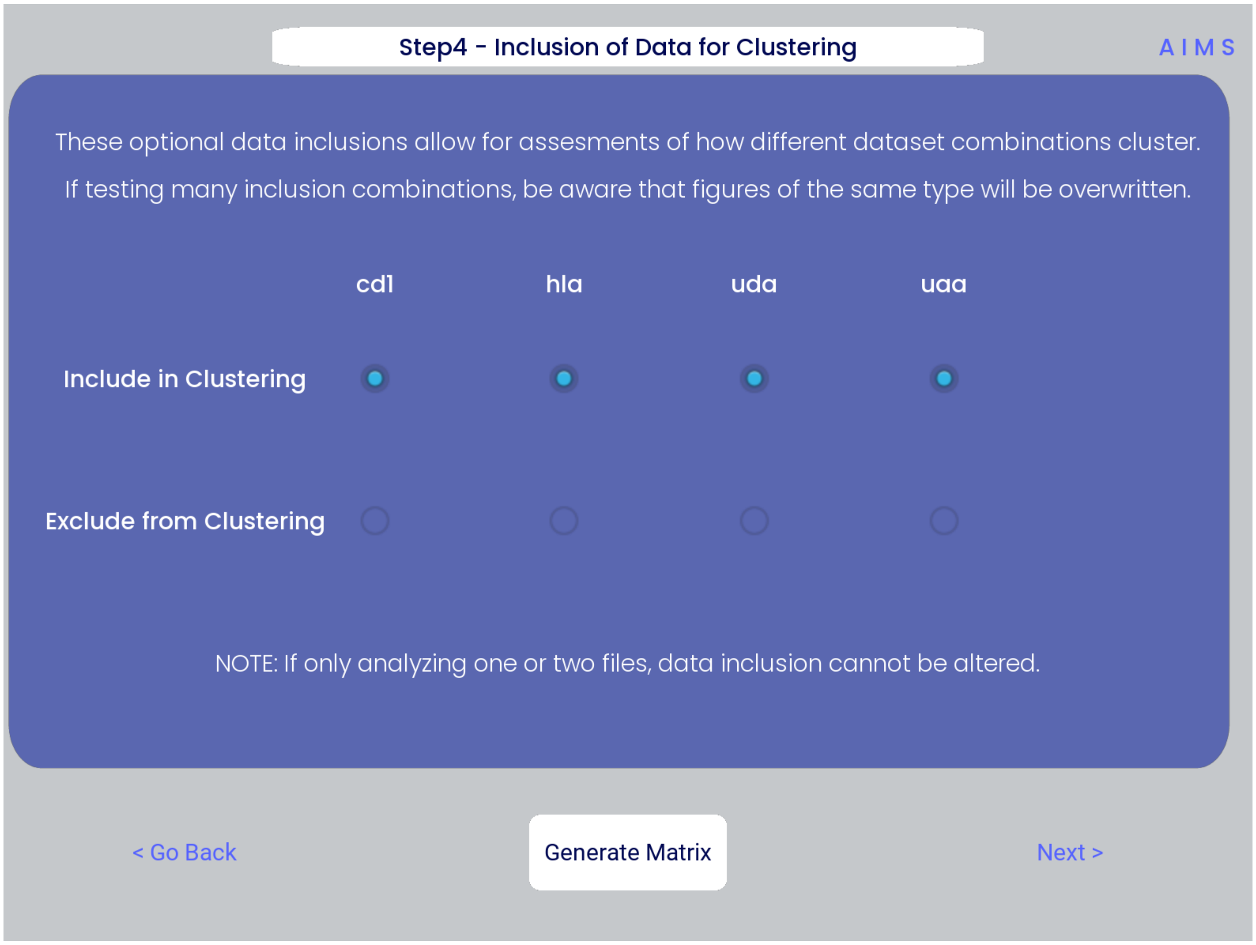
Horizontal white lines separate data from files loaded in previous step.

Congrats! You've generated your first piece of data using this software. You might notice that your image quality is poor for figures shown in the app, this is because the software shows *png files. Don't worry, both a *png and a higher-quality *pdf version of the plot are saved in whichever directory you specified in Step 2. This is true for subsequently generated figures.

Additionally, as you move through the sequential steps of the GUI, keep in mind that **all generated figures have corresponding raw data saved to a *.dat file**, should the user want to re-plot the data using a different color scheme, different plotting application, etc. For screen 3, the output figures are matrix.png and matrix.pdf, while the ouput raw data is saved as raw_matrix.dat.

Step 4: Generate High-Dimensional Biophysical Matrix

In this step, we generate the high-dimensional biophysical property matrix that will be used in all downstream analysis. We then have the option to include or exclude files from the clustering that will happen in the next step. If only one or two datasets are included in the analysis, all input data must be included in the clustering. Again, we simply press the "Generate Matrix" button, shown below, and then users can move on to the next step.



Step4 - Inclusion of Data for Clustering

These optional data inclusions allow for assesments of how different dataset combinations cluster. If testing many inclusion combinations, be aware that figures of the same type will be overwritten.

	cd1	hla	uda	uaa
Include in Clustering	●	●	●	●
Exclude from Clustering	○	○	○	○

NOTE: If only analyzing one or two files, data inclusion cannot be altered.

Note: Don't worry if this step takes a little while, especially for larger datasets. This will be the slowest and most memory-intensive step in the analysis.

While most users may not want to leave any datasets out of the clustering of Step 5, there are some interesting applications of AIMS where the exclusion of certain datasets has interesting impacts on receptor clustering. It is important to remember that the UMAP and PCA dimensionality reduction steps are strongly dependent on the input data. Learn more about this input data dependence in the *AIMS Cluster* section.

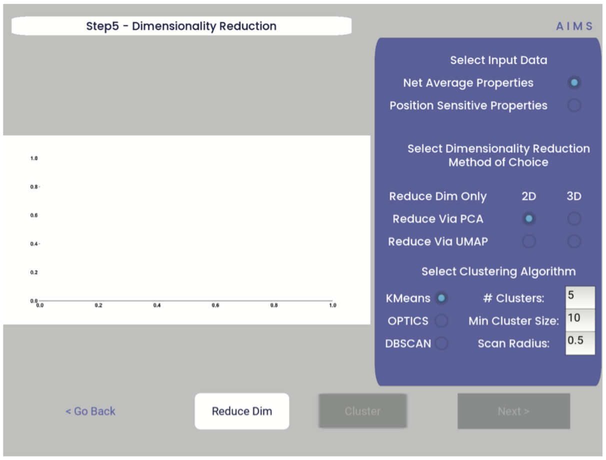
Step 5: Dimensionality Reduction and Receptor Clustering

The goal in this step is to take that large biophysical property matrix generated in the previous step, and reduce this high-dimensional matrix down to two or three composite dimensions, and then cluster the receptors projected onto this space based upon distance in this projected space. This step is perhaps the most involved in the GUI, with the most customizable options. Hence the dedicated *AIMS Cluster* section detailing the possibilities for this analysis.

First, the user must decide if they would like to reduce dimensionality on Net Average Properties, i.e. the biophysical properties averaged across entire receptors, or on the Position Sensitive Properties, i.e. the amino acid biophysical properties at every position on every receptor.

Next, the algorithm used for this dimensionality reduction must be chosen. Users can choose either Principal Component Analysis (PCA) or Uniform Manifold Approximation and Projection (UMAP), and additionally choose to visualize these projections in two- or three-dimensions. Once these options are chosen, click the “Reduce Dim” button to visualize these options. More options can be tested and the projection re-visualized as many times as the user desires.

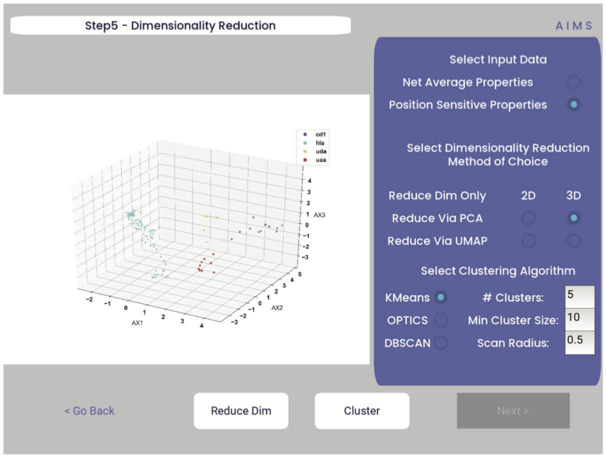
Lastly, the data is then clustered using one of three algorithms, either K-Means, OPTICS, or DBSCAN clustering. Users must also define, for each of these algorithms, a tunable parameter that determines the size of the clusters generated. We can see each of these options, and the default values for the tunable parameters, in the screenshots below.


Step5 - Dimensionality Reduction

Select Input Data
Net Average Properties ●
Position Sensitive Properties ○

Select Dimensionality Reduction Method of Choice
Reduce Dim Only 2D 3D
Reduce Via PCA ● ○
Reduce Via UMAP ○ ○

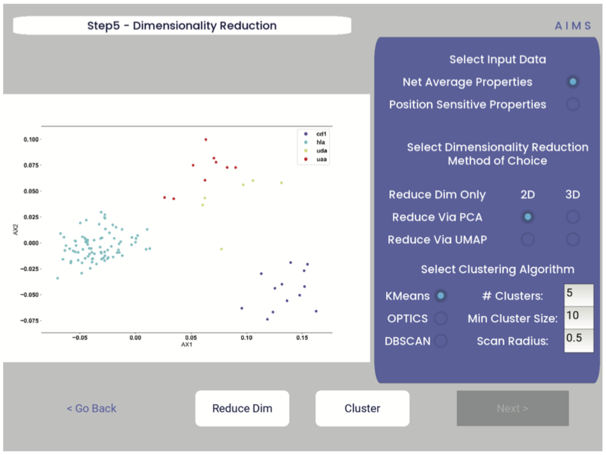
Select Clustering Algorithm
KMeans ● # Clusters: 5
OPTICS ○ Min Cluster Size: 10
DBSCAN ○ Scan Radius: 0.5


Step5 - Dimensionality Reduction

Select Input Data
Net Average Properties ○
Position Sensitive Properties ●

Select Dimensionality Reduction Method of Choice
Reduce Dim Only 2D 3D
Reduce Via PCA ○ ●
Reduce Via UMAP ○ ○

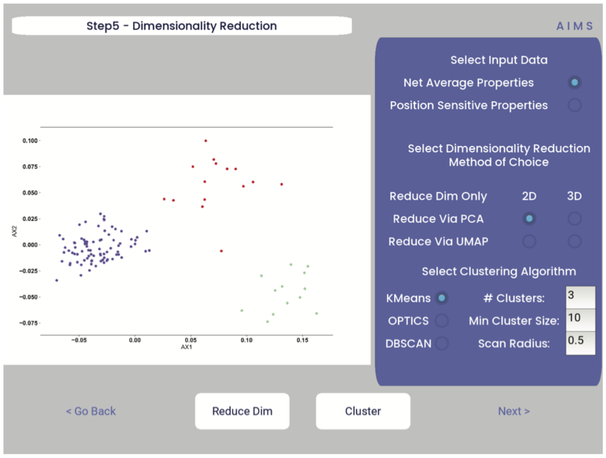
Select Clustering Algorithm
KMeans ● # Clusters: 5
OPTICS ○ Min Cluster Size: 10
DBSCAN ○ Scan Radius: 0.5


Step5 - Dimensionality Reduction

Select Input Data
Net Average Properties ●
Position Sensitive Properties ○

Select Dimensionality Reduction Method of Choice
Reduce Dim Only 2D 3D
Reduce Via PCA ● ○
Reduce Via UMAP ○ ○

Select Clustering Algorithm
KMeans ● # Clusters: 5
OPTICS ○ Min Cluster Size: 10
DBSCAN ○ Scan Radius: 0.5


Step5 - Dimensionality Reduction

Select Input Data
Net Average Properties ●
Position Sensitive Properties ○

Select Dimensionality Reduction Method of Choice
Reduce Dim Only 2D 3D
Reduce Via PCA ● ○
Reduce Via UMAP ○ ○

Select Clustering Algorithm
KMeans ● # Clusters: 3
OPTICS ○ Min Cluster Size: 10
DBSCAN ○ Scan Radius: 0.5

For more detail on how these dimensionality reduction and clustering algorithms work, as well as details on the tunable parameters, please see the *AIMS Cluster* section.

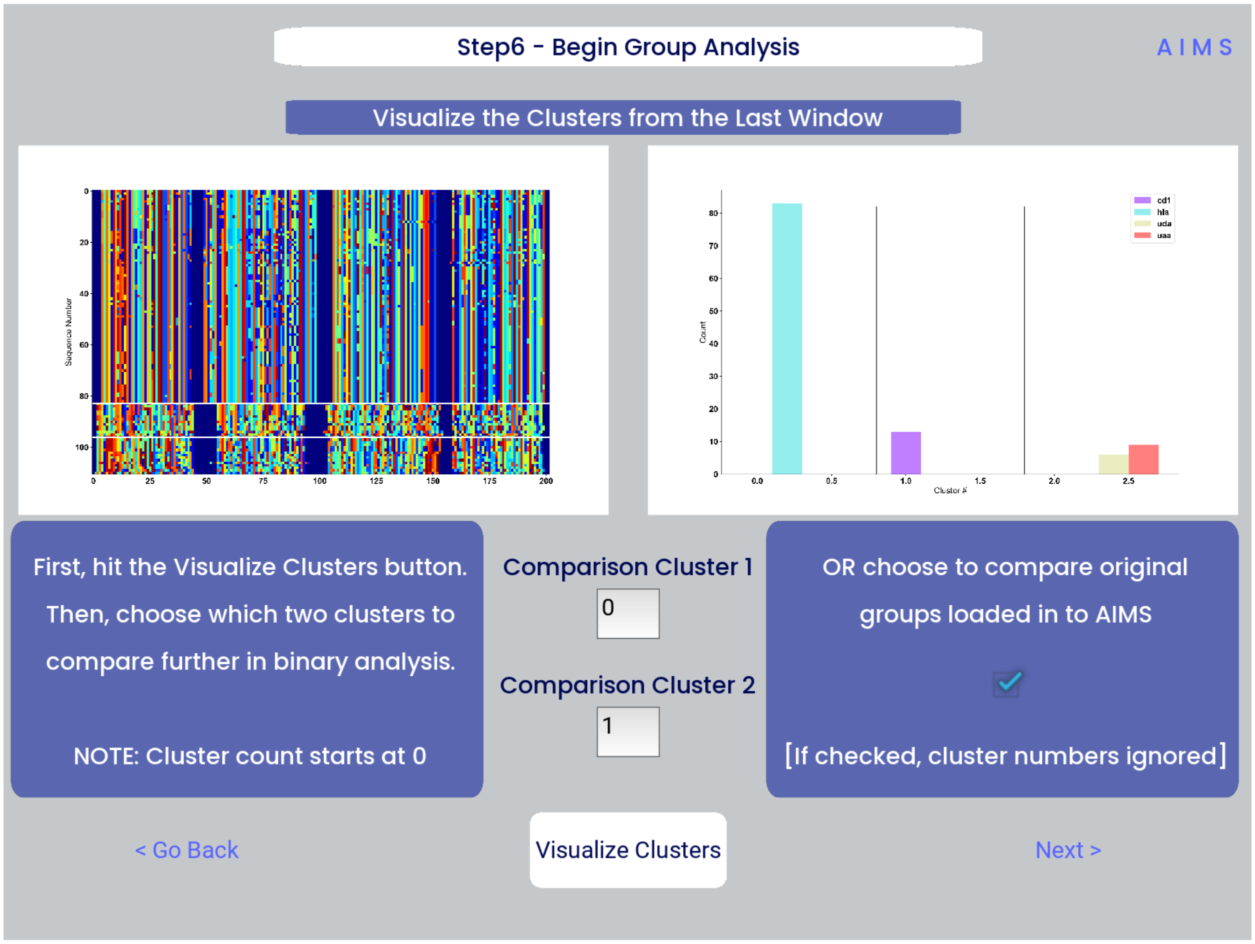
In the above screenshots, we see first the default screen (top left), then the three-dimensional PCA projection (top right), followed by a Kmeans clustering with 5 clusters (bottom left), and lastly an OPTICS clustering with a minimum cluster size of 5 (bottom right). Users should note that Kmeans will cluster all sequences in the dataset, while OPTICS and DBSCAN will exlude sequences that are not found at a sufficient density in the projection. These unclustered sequences are grayed out in the resultant displayed figure.

There is no one right answer to determining the “best” dimensionality reduction or clustering algorithm, so users are encouraged to try a range of options to determine which combination makes the most sense for their data. Importantly for this step, generated figures and data from each dimensionality reduction and clustering algorithm are **not overwritten**. You will see in the output directory descriptive filenames that correspond to each option used. Importantly however, for each given clustering algorithm only one file will be saved. If, for instance, you cluster your data using the Kmeans algorithm with “# Clusters” set to 5, then run it again with “# Clusters” set to 10, the output figures and data will reflect only the “# Clusters = 10” configuration. Lastly, raw data outputs “umap_3d.dat” or “pca_3d.dat” give the location of each sequence in the reduced dimensionality projection. Raw data outputs for the clustering steps “optics_3d.dat” or “dbscan_3d.dat” reflect the cluster membership of each given sequence, with the order of sequences preserved based upon your input datasets.

Note: Whichever projection and clustering algorithm the user is currently viewing when moving to the next step is what will be used in all downstream analysis

Step 6: Visualize and Analyze Clustered Sequences

At this stage, we visualize the clustered sequences from the previous step. First, the user must hit the "Visualize Cluster" button, then after visual inspection of the clusters and sequences, the "comparison clusters" can be selected. The goal of this step is to determine whether the user wants to compare two biophysically distinct clusters which were identified in the previous step, or compare across the input datasets. We can see in the screenshot below how this works:



Step6 - Begin Group Analysis

First, hit the Visualize Clusters button. Then, choose which two clusters to compare further in binary analysis.

NOTE: Cluster count starts at 0

Comparison Cluster 1: 0

Comparison Cluster 2: 1

OR choose to compare original groups loaded in to AIMS

☑ [If checked, cluster numbers ignored]

After the cluster visualization is complete, we see in the right panel, left figure that the matrix from step 3 is rearranged to reflect the clustered sequences, with higher sequence conservation (colors in the matrix) evident within each cluster. In the right figure, we see the sequence count of each input dataset in each cluster.

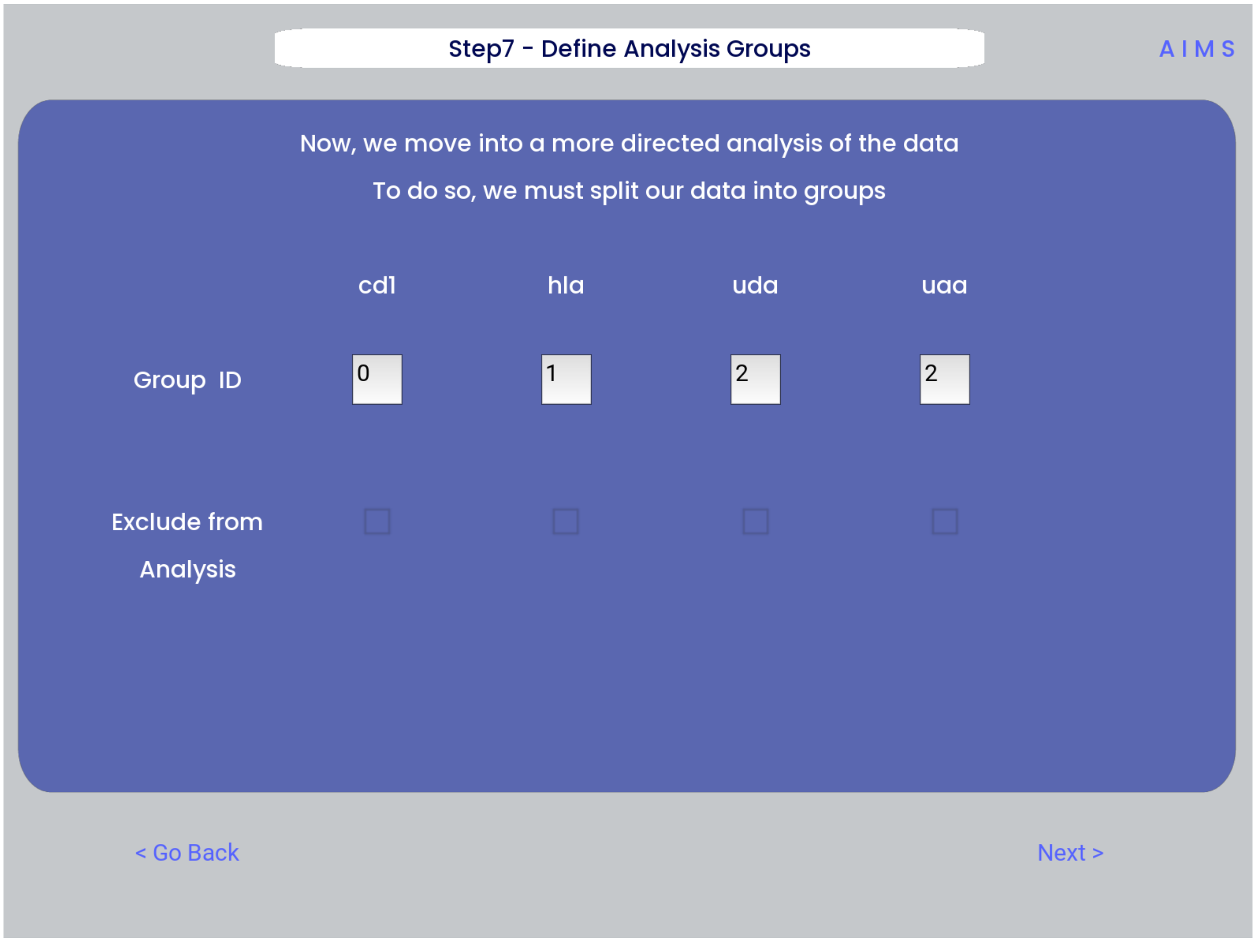
From this information, the user can determine which clusters they would like to analyze by entering values in the "Comparison Cluster" boxes. The cluster count starts at zero, and the user can infer the last cluster number from the figure on the right. The amino acid sequences from the selected clusters will be saved to the output directory as "clust2seq_#.txt" where the "#" is the cluster number for each respective dataset.

If the user instead is still most interested in comparing the input datasets, the checkbox on the right side of the screen can be checked, ignoring the clustering of the data (but still saving the results in the output directory!).

Warning: The clustering algorithms are stochastic, and so cluster ID and cluster membership may change each time the software is run. For instance, in this walkthrough I use clusters 10 and 11 for downstream analysis, but users trying to replicate this analysis may have different sequences in clusters 10 and 11. This is important both for comparisons in this walkthrough as well as creating reproducible analysis.

Step 7: Define Comparison Classes

Here, we separate our loaded data into separate classes for downstream analysis, assuming the user opted not to compare clustered sequences. As a default, each loaded dataset is assigned to its own unique group, but the user may group these datasets however they choose by assigning matching group numbers to datasets they want analyzed together.



Step7 - Define Analysis Groups

Now, we move into a more directed analysis of the data

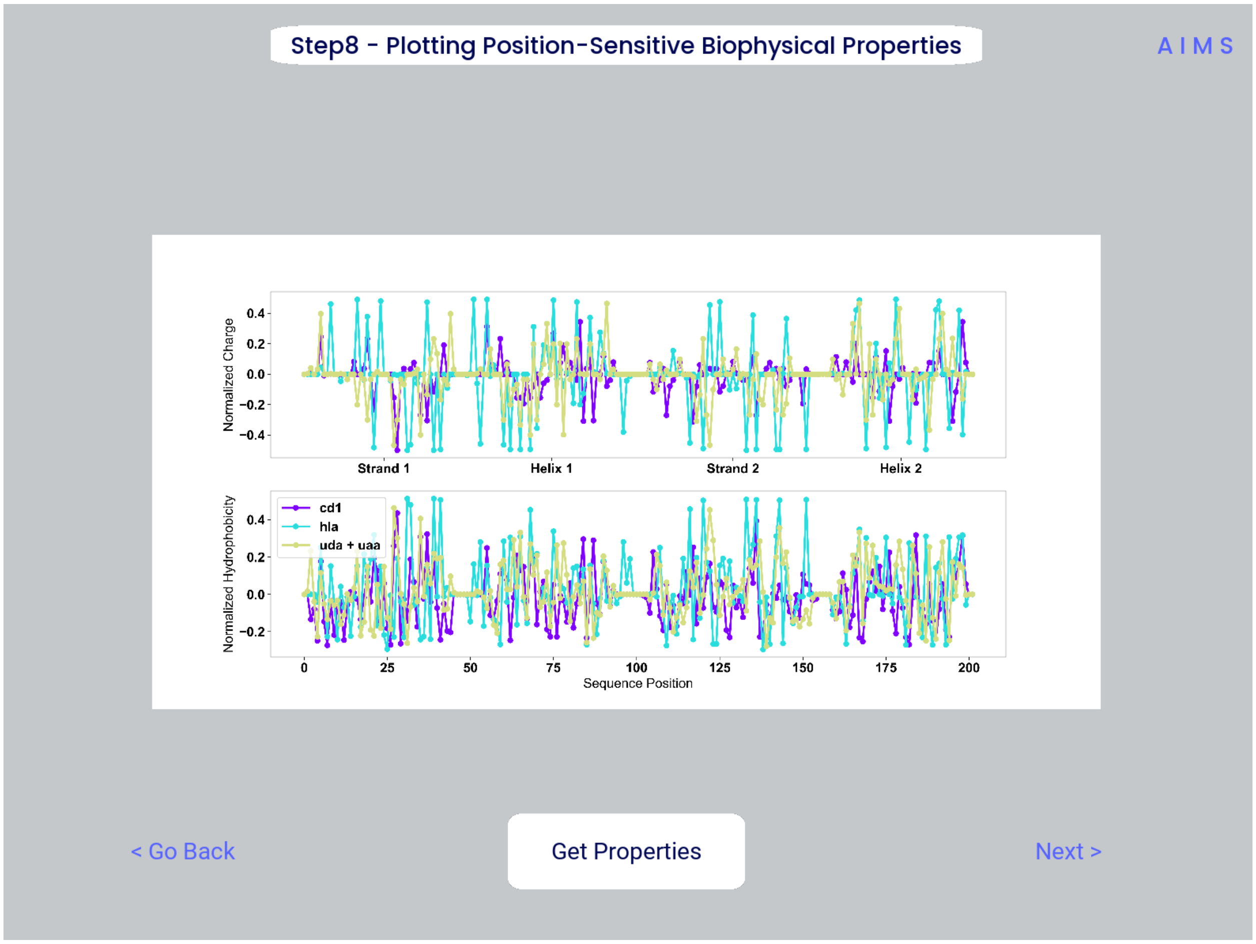
To do so, we must split our data into groups

	cd1	hla	uda	uaa
Group ID	0	1	2	2
Exclude from Analysis	□	□	□	□

Warning: If comparing more than two distinct groups, some of the analysis will be unavailble. These analyses include mutual information analysis, amino acid frequency characterization, and linear discriminant analysis. Each of these analyses require binary classifications of the data.

Step 8: Visualize Averaged Position Sensitive Biophysical Properties

In this step we look at average biophysical properties as a function of sequence space, part of our special "positional encoding". At this stage in the walkthrough we won't bother showing the "before" snapshots of the GUI, as the only options are to press the button which generates the plot, and then move on to the next step. However, if you're trying to compare the results to the data we get in this walkthrough, the generated plots are quite useful:


Step8 - Plotting Position-Sensitive Biophysical Properties
A I M S
Normalized Charge
0.4
0.2
0.0
−0.2
−0.4
Strand 1
Helix 1
Strand 2
Helix 2
Normalized Hydrophobicity
cd1
hla
uda + uaa
0.4
0.2
0.0
−0.2
0
25
50
75
100
125
150
175
200
Sequence Position
< Go Back
Get Properties
Next >

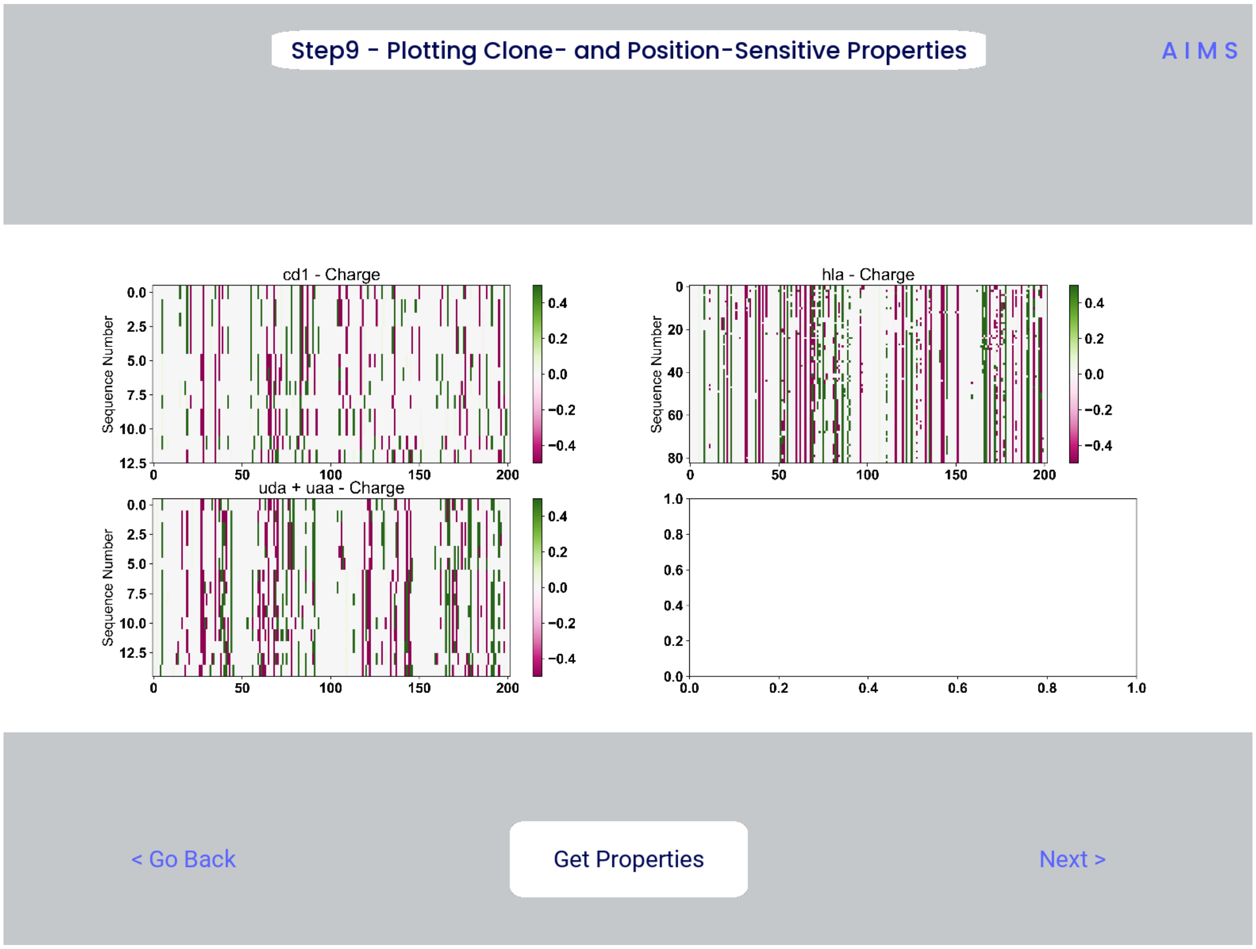
Note: Standard deviations are not shown, and ideally these would be calculated via bootstrapping

The figure in this step is saved as “pos_prop.pdf/png”, while the raw data is saved as “position_sensitive_mat#.dat” where the “#” again corresponds to the selected cluster number, or if comparing the original input datasets, the user-defined group number. This data file has as many rows as the number of sequences in the selected cluster or group, and has 61 x # AIMS positions columns. Ideally this data would be saved as a tensor of shape # sequences x 61 x # AIMS positions, however, this would require saving the data as a numpy object, which would be less friendly to other programming languages for replotting and reformatting. The 61 here refers to the 61 biophysical properties of AIMS (listed in the *Biophysical Properties*).

As a concrete example, cluster 10 in this walkthrough has 10 sequences, and 95 AIMS positions. So the first row of the position_sensitive_mat10.dat file here corresponds to the first sequence (which can be found in the “clust2seq_10.dat” output). The first 95 columns in this row correspond to the position sensitive charge (biophysical property 1 of 61). The next 95 columns correspond to the position sensitive hydrophobicity (biophysical property 2 of 61). And so on.

Step 9: Visualize Raw Position Sensitive Biophysical Properties

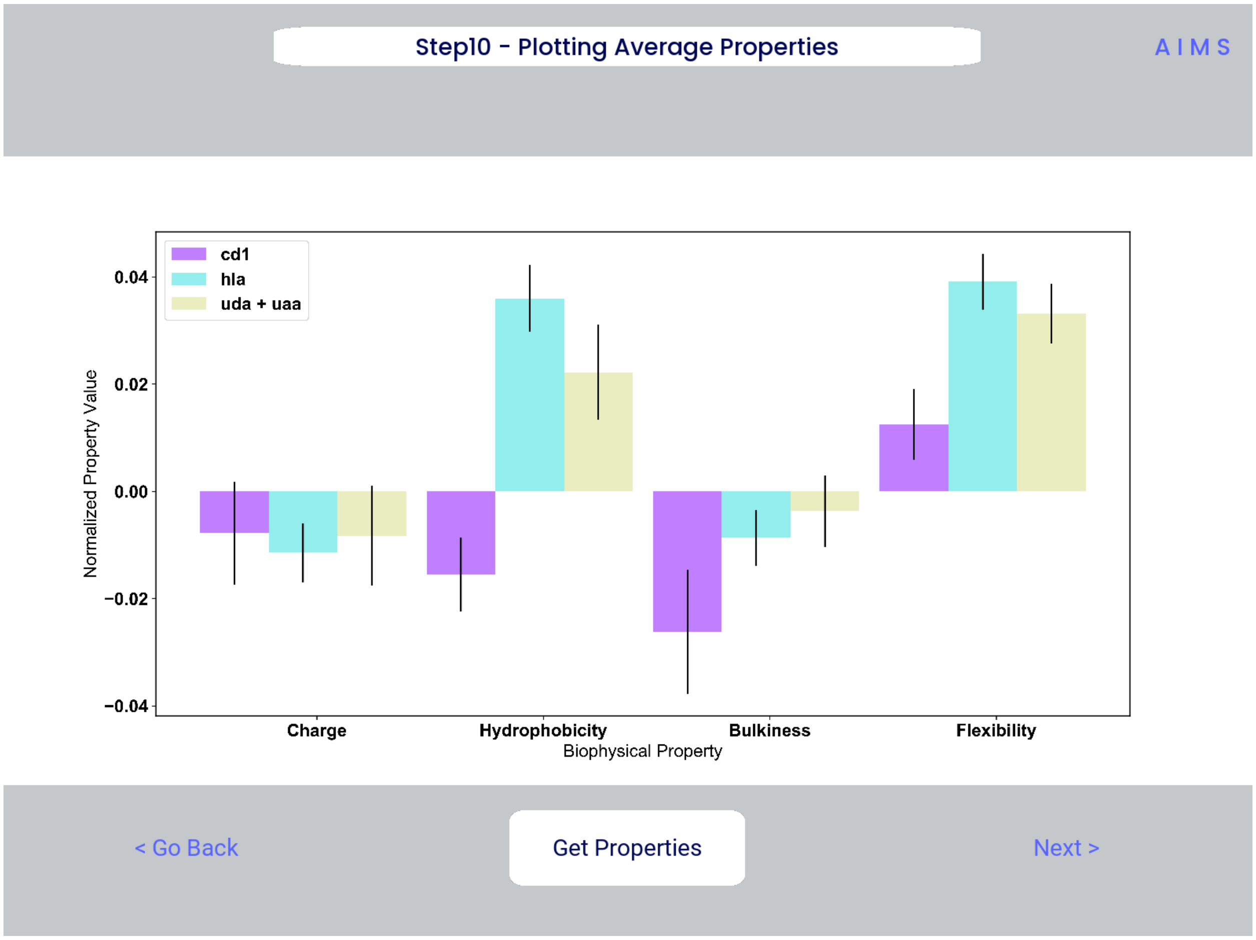
In this step, we visualize the position sensitive charge for all clones, not averaged. This figure can help provide a sense of how reliable the averages of the previous step are. Like all biophysical properties in AIMS, the charge is normalized, hence the minimum and maximum on the scales not equaling 1.



These figures are saved as “clone_pos_prop.pdf/png”. This figure helps to understand the figure generated in Step 8, which is simply this figure averaged over the y-axis. It is additionally important to note that the positional encoding in Steps 8 and 9 are consistent with the alignment scheme selected in Step 3, either central, bulge, right, or left aligned.

Step 10: Visualize Net Biophysical Properties

In this step, we are averaging the biophysical properties over all positions and all receptors. In other words, effectively averaging the figures generated in Step 9 over both the x- and the y-axes.



Note: A large standard deviation in these plots are to be expected, especially if users are analyzing original input datasets rather than selected cluster subsets

This figure is saved as “avg_props.pdf/png”, while statistical significance, as calculated using Welch’s t-Test with the number of degrees of freedom set to 1, is saved as “avg_prop_stats.csv”.

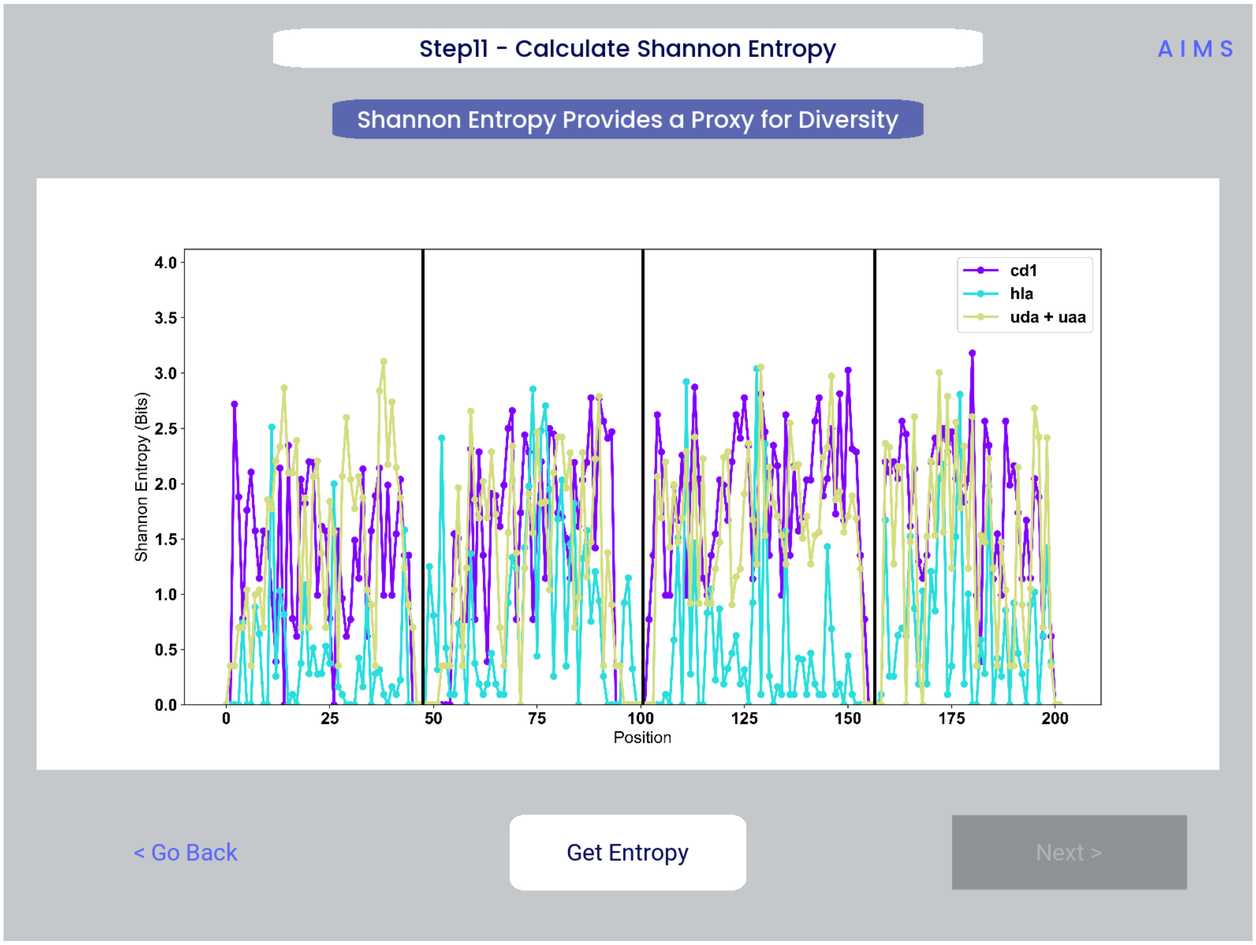
Step 11: Calculate Shannon Entropy

In this step, we are calculating the Shannon Entropy of the chosen datasets, effectively the diversity of the receptors as a function of position. For more information on the Shannon Entropy, as well as the Mutual Information discussed in the next step, view the Information Theory section of the *Core Functionalities*.

This figure is saved as “shannon.pdf/png”.

Note: Due to the requirement for a binary comparison in subsequent steps, this is the last GUI screen if users are comparing more than 2 groups

**END MHC Analysis **



Step11 - Calculate Shannon Entropy

Shannon Entropy Provides a Proxy for Diversity

Due to the analysis of three distinct groups in this walkthrough, the MHC analysis ends here at step 11. However, as discussed previously these steps are identical to those in the immunoglobulin analysis. If you'd like to learn more about steps 12, 13, and 14, go back to the end of the *Immunoglobulin Analysis with AIMS*.

Otherwise, congratulations on completing the AIMS MHC walkthrough! Thanks for using the software, and be sure to reach out if there are any outstanding questions/unclear sections of this walkthrough.

1.3 AIMS Basics

In this section, we will focus on the absolute basics that you need to know about while running AIMS. This includes the *Input Formatting* of the files used for the analysis, the *Core Functionalities* of the software, and the *Biophysical Properties* that are the central pillars of the analysis. This information is key whether users are interested in working with the AIMS GUI or with the AIMS Jupyter Notebooks. The hope is that if there are any fundamental questions with the meaning of a given output, or necessary troubleshooting with inputs, the users can find them here.

1.3.1 Input Formatting

At present, the AIMS software requires a specific input formatting to be read by the software. Future versions of the software will hopefully have more relaxed formatting requirements, assuming users request such functionality. Here we will go into specifics for each molecular species, but the general format maintained for all species involves a comma separated value (CSV) file with a header in the first row, and no associated metadata in the given file.

Immunoglobulin (TCR/Ab) Formatting

The immunoglobulin (Ig) input file formatting is the most flexible of the three, with options meant to satisfy the needs of any given AIMS input data. Specifically these data must contain only the complementarity determining region (CDR) loops, as AIMS excludes the framework regions of antibodies and TCRs in the analysis. An example of the proper input formatting for the first few lines of an input file can be seen below:

```
[Example file from AIMS/app/ab_testData/flu_poly.csv]
l1,l2,l3,h1,h2,h3
QSISSY,DAS,QHRSTWPPN,GGTFSSRA,IIPIFNTP,AREMATIFGRMDV
ESLLHSDGKTY,EVS,MQTIQLPGT,GGIMRRNG,IIAIFGTP,VASSGYHLHRETWGY
QDIKNY,HVS,HQCYNLPYT,GFIFGHFA,ISGGGLNT,ARFDSSGYNYVRGMVV
.
.
.
```

The key features are that 1. the general format must follow that of a comma separated value (csv) file. In this csv file each row represents a unique sequence, each column represents a given CDR loop, and each column is separated by a comma. 2. Each column must have a header with no sequence information in it. The contents of this header are not critical, as the standard AIMS Ig file loader disregards this header and replaces it. Descriptive headers are preferred, if only for downstream transparency of the data. 3. Single letter amino acid codes should be used, with only capitalized letters. The defined function "aimsLoad.convert_3let" can convert three letter codes to single letter codes [see API for more details… once I make this section]. 4. The sequences do not have any extranneous characters or spaces. Currently AIMS is capable of identifying an "X" in a sequence and by default removes these sequences. Any other non-amino acid single letter characters will result in an error, and spaces will be encoded into the AIMS matrix, which could confound analysis.

At present, AIMS does NOT identify other issues with sequences. Missing CDR loops in a sequence, spaces included in a sequence, or other miscellaneous mistakes will not result in an error, and could lead to inaccuracies in the downstream analysis. Either visual inspection or other quality control steps should be taken to ensure proper analysis. If analyzing multiple files at once, all input files must have the same number of loops.

Note: Currently, AIMS cannot analyze TCR/Ab inputs of 4 or 5 loops. For most repertoire analysis, the requirement to analyze such a dataset would be unexpected. Users should submit an issue on the GitHub if this analysis is needed for some reason.

MHC/MHC-Like Formatting

The current MHC/MHC-like analysis requires the most rigid and restrictive of all the input formatting, but this will hopefully change in future updates. Users can either input entire MHC sequences or just the platform domain sequences. Additionally, users can leverage these restrictive requirements to analyze other molecular species if needed. The initial input is a simple aligned FASTA, as seen below:

```
[Example file from AIMS/app/mhc_testData/hlaA_seqs.fasta]
>3VJ6_A Chain A, H-2 Class I Histocompatibility Antigen, D-37 Alpha Chain [Mus musculus]
------------------------------------------------------------
------------------------------------------------------------
-------------------------------------------------SPHSLRYFTTA
VSRPGLGEPRFIIVGYVDDTQFVRFDSDAENPRMEPRARWIEQEGPEYWERETWKAR
DMGRNFRVNLRTLLGYYNQSNDESHTLQWMYGCDVGPDGRLLRGYCQEAYDGQDYISLNE
DLRSWTANDIASQISKHKSEAVDEAH-QQRAYLQGPCVEWLHRYLRLGNETLQRSDPPKA
HVTHHPRSEDEVTLRCWALGFYPADITLTWQLNGEELTQDMELVETRPAGDGTFQKWAAV
VVPLGKEQYYTCHVYHEGLPEPLTLRWEPP------------------------------
-------------------------------------------------
>5VCL_A Chain A, H2-t23 Protein [Mus musculus]
------------------------------------------------------------
------------------------------------------------------------
------------------------------------------------MSSHSLRYFHTA
.
.
.
```

Importantly, each FASTA entry must be pre-aligned using BLAST or a similar alignment software. AIMS does not internally align the sequences, and requires that the inputs can be expected to be structurally very similar. For MHC and MHC-like molecules, this requirement is satisfied. If a subset of sequences align poorly for some reason, they can be included as a separate file. Each individual file will have its own user-specified region of the alignment that will ultimately be input into the analysis. The user specification can be done on-the-fly, or input as a separate file formatted as such:

```
[Example file from AIMS/app/mhc_testData/ex_cd1_hla_uda_uaa.csv]
Name,S1s,S1e/H1s,H1e/S2s,S2e/H2s,H2e
cd1,124,167,209,262,303
hla,170,210,260,306,348
uda,2,49,93,152,193
uaa,2,49,93,152,193
```

The above file is formatted again as a comma separated value (csv), with the first column giving the name of the dataset, and the remaining columns identifying the start and end point of four distinct structural features in the provided FASTA alignment. Specifically for the analysis of MHC and MHC-like molecules, these four structural features are the beta-strand of the alpha 1 domain, the alpha helix of the alpha 1 domain, the beta-strand of the alpha 2 domain, and the alpha helix of the alpha 2 domain. Each number represents either the start of one structural feature, the end of another structural feature, or both. In the example file, for the hla alignmemnt (corresponding to the FASTA above) the first beta strand starts at alignment position 170 and ends at position 210. Likewise, the first alpha helix starts at position 210 and ends at position 260. And so on.

Currently, the Phyre server (http://www.sbg.bio.ic.ac.uk/phyre2/html/page.cgi?id=index) is recommended to identify these structural features. Other software may be used to identify the key structural features for analysis, but the numbering provided in standard Phyre outputs makes translation to the above csv file easy. Generally only one sequence should be necessary to be used as input, as structural similarity is a requirement for comparable analysis using AIMS. Users can take advantage of this ambiguity in the software to analyze any four connected structual features in evolutionarily and structurally related molecules in the AIMS GUI. Users comfortable with the Jupyter Notebooks can instead follow the Multi-Sequence Alignment formatting instructions.

Immunopeptidomics Formatting

This is the first of two sections that are not yet implemented in the AIMS GUI, but can be analyzed using the AIMS notebooks. Specifically, the AIMS_peptide.ipynb file can be used to analyze immunopeptidomics data. Again the input is simply a comma separated value (csv) formatted file. However, since the input should only have one column, the precise format is a little less important. An example can be seen below:

```
[Example file from AIMS/app_data/pancreas_hla_atlas.csv]
sequence
ALVSGNNTVPF
TYRGVDLDQLL
NYIDIVKYV
SYIPIFPQ
NYFPGGVALI
.
.
.
```

Example data provided from the HLA Ligand Atlas (https://hla-ligand-atlas.org/welcome). In future releases, data related to mass spectrometry approaches used for the identification of these peptides will be included in the analysis. Metadata will be included in additional columns of the csv.

Multi-Sequence Alignment Formatting

Again, this multi-sequence alignment input is not yet available in the AIMS GUI, is available in the notebook. Unlike the MHC input formatting described above, structural features should be identified before being loaded into AIMS. Each entry should have the exact same number of characters, and an associated sequence name, as seen below:

```
[Example file from AIMS/app/MSA_testData/Dpr_interface.csv]
Dpr,Sequence
Dpr1,DKDVSWIRKRDLHILTAGGTTYTSD-----QINTEPKMSLSYTFNVVEL
Dpr2,DKSVSWIRKRDLHILTAGILTYTSD-----QVNTEPKISMAFRLNVIVT
Dpr3,DKSVSWIRKRDLHILTVGTATYTSD-----QVNTEPKMSMAFQLNIIEI
.
.
.
```

The required sequence name is used in the downstream analysis in this case, to help identify potential clusters of biophysically similar sequences. Unlike in the analysis of TCR, MHC, or peptide sequences, the associated names or identifiers for each sequence are more likely to be clinically or evolutionarily important.

1.3.2 Core Functionalities

Functionalities coming soon!

1.3.3 Biophysical Properties

In generating the core biophysical property matrix of the AIMS analysis, the same 61 biophysical properties are used in all analyses, with an option to use fewer if the user decides to. The properties are listed in the table below:

Table 1: Table of AIMS Biophysical Properties

Number	Property [Shorthand]	Decription
0	Hydrophobicity1 [Phob1]	Hydrophobicity Scale [-1,1]
1	Charge [Charge]	Charge [ec]
2	Hydrophobicity2 [Phob2]	Octanol-Interface Hydrophobicity Scale
3	Bulkiness [Bulk]	Side-Chain Bulkiness
4	Flexibility [Flex]	Side-Chain Flexibility
5	Kidera 1 [KD1]	Helix/Bend Preference
6	Kidera 2 [KD2]	Side-Chain Size
7	Kidera 3 [KD3]	Extended Structure Preference
8	Kidera 4 [KD4]	Hydrophobicity
9	Kidera 5 [KD5]	Double-bend Preference
10	Kidera 6 [KD6]	Flat Extended Preference
11	Kidera 7 [KD7]	Partial Specific Volume
12	Kidera 8 [KD8]	Occurrence in alpha-region
13	Kidera 9 [KD9]	pK-C
14	Kidera 10 [KD10]	Surrounding Hydrophobicity
15	Hotspot 1 [HS1]	Normalized Positional Residue Freq at Helix C-term
16	Hotspot 2 [HS2]	Normalized Positional Residue Freq at Helix C4-term
17	Hotspot 3 [HS3]	Spin-spin coupling constants
18	Hotspot 4 [HS4]	Random Parameter
19	Hotspot 5 [HS5]	pK-N
20	Hotspot 6 [HS6]	Alpha-Helix Indices for Beta-Proteins
21	Hotspot 7 [HS7]	Linker Propensity from 2-Linker Dataset
22	Hotspot 8 [HS8]	Linker Propensity from Long Dataset
23	Hotspot 9 [HS9]	Normalized Relative Freq of Helix End
24	Hotspot 10 [HS10]	Normalized Relative Freq of Double Bend
25	Hotspot 11 [HS11]	pK-COOH
26	Hotspot 12 [HS12]	Relative Mutability
27	Hotspot 13 [HS13]	Kerr-Constant Increments
28	Hotspot 14 [HS14]	Net Charge
29	Hotspot 15 [HS15]	Norm Freq Zeta-R
30	Hotspot 16 [HS16]	Hydropathy Scale
31	Hotspot 17 [HS17]	Ratio of Average Computed Composition
32	Hotspot 18 [HS18]	Intercept in Regression Analysis
33	Hotspot 19 [HS19]	Correlation coefficient in Reg Anal
34	Hotspot 20 [HS20]	Weights for Alpha-Helix at window pos
35	Hotspot 21 [HS21]	Weights for Beta-sheet at window pos -3
36	Hotspot 22 [HS22]	Weights for Beta-sheet at window pos 3

continues on next page

Table 1 – continued from previous page

Number	Property [Shorthand]	Decription
37	Hotspot 23 [HS23]	Weights for coil at win pos -5
38	Hotspot 24 [HS24]	Weights coil win pos -4
39	Hotspot 25 [HS25]	Weights coil win pos 6
40	Hotspot 26 [HS26]	Avg Rel Frac occur in AL
41	Hotspot 27 [HS27]	Avg Rel Frac occur in EL
42	Hotspot 28 [HS28]	Avg Rel Frac occur in A0
43	Hotspot 29 [HS29]	Rel Pref at N
44	Hotspot 30 [HS30]	Rel Pref at N1
45	Hotspot 31 [HS31]	Rel Pref at N2
46	Hotspot 32 [HS32]	Rel Pref at C1
47	Hotspot 33 [HS33]	Rel Pref at C
48	Hotspot 34 [HS34]	Information measure for extended without H-bond
49	Hotspot 35 [HS35]	Information measure for C-term turn
50	Hotspot 36 [HS36]	Loss of SC hydropathy by helix formation
51	Hotspot 37 [HS37]	Principal Component 4 (Sneath 1966)
52	Hotspot 38 [HS38]	Zimm-Bragg Parameter
53	Hotspot 39 [HS39]	Normalized Freq of ZetaR
54	Hotspot 40 [HS40]	Rel Pop Conformational State A
55	Hotspot 41 [HS41]	Rel Pop Conformational State C
56	Hotspot 42 [HS42]	Electron-Ion Interaction Potential
57	Hotspot 43 [HS43]	Free energy change of epsI to epsEx
58	Hotspot 44 [HS44]	Free energy change of alphaRI to alphaRH
59	Hotspot 45 [HS45]	Hydrophobicity coeff
60	Hotspot 46 [HS46]	Principal Property Value z3 (Wold et. al. 1987)

The so-called Kidera factors are from the published work:

Kidera et al. Statistical analysis of the physical properties of the 20 naturally occurring amino acids. Journal of Protein Chemistry (1985)

While the hotspot variables mentioned above are from:

Liu et al. Hot spot prediction in protein-protein interactions by an ensemble system. BMC Systems Biology (2018)

1.4 AIMS Cluster

As discussed in the *Core Functionalities*, the key step in the AIMS analysis pipeline is the generation of a high-dimensional biophysical property matrix based upon the initial AIMS position-sensitive sequence encoding. This high-dimensional matrix provides an exceptional opportunity to identify biophysically similar clusters of receptors. The two steps in generating these clusters are discussed below in the *Dimensionality Reduction* and *Picking the Proper Clustering Algorithm* sections.

A key concept of this entire page is that there is no single "correct" way to reduce dimensionality and cluster a given dataset. The type of data being analyzed and the information users are trying to glean from this data will be key for determining the best combination of approaches. Users are encouraged to try multiple approaches, and glean information about the data from each approach.

1.4.1 Dimensionality Reduction

In order to properly cluster our sequences and visualize these clusters for user interpretation, the high-dimensional biophysical property matrices must be projected onto a lower dimensional space. AIMS offers three options for generating this projection while best preserving the key features of the high-dimensional biophysical property space.

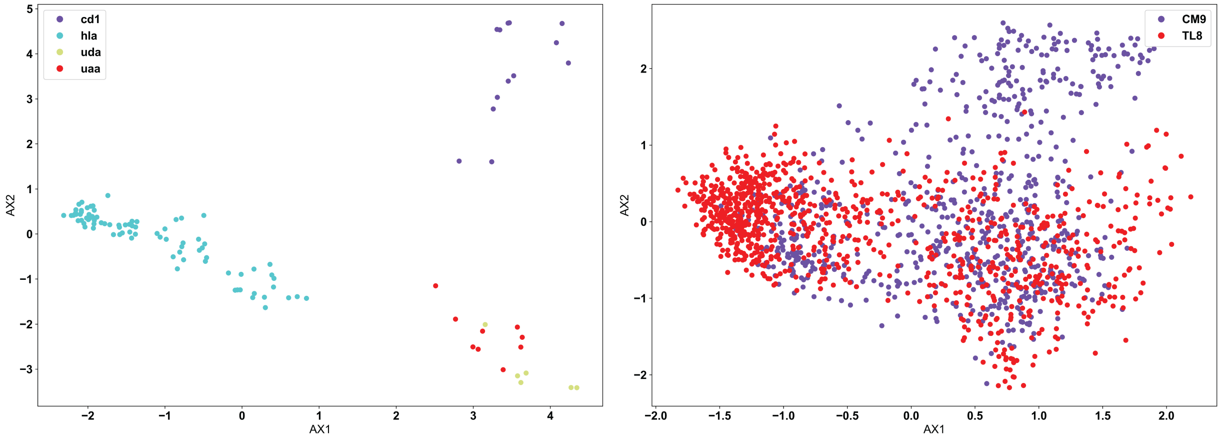
Principal Component Analysis

The preferred dimensionality reduction technique utilized in AIMS is principal component analysis (PCA). As a linear and deterministic analysis approach, PCA is the most interpretable and reproducible approach. The principal components identified give back the dimensions of the data with the highest variance in the dataset. Each principal component is an orthonormal vector spanning this dimension of highest variance. So, in a way, you can consider the most distal sequences in the projected PC space to be the most "biophysically distinct" within your dataset.

Regarding this linearity, the principal components themselves are simply linear combinations of the input biophysical properties. So, if your biophysical property matrix has 90 vectors (columns) for each sequence, then the principal components will weight all 90 of these vectors by some factor. Each individual sequence is then projected onto these principal components via a simple sum. So for instance, if principal component 1 (PC1) has weights 0.3 - Charge, 0.15 - Phob1, 0.05 - Bulk, etc. then the resultant projection of sequence 1 onto PC1 will be 0.3*Seq1_charge + 0.15*Seq1_phob1 + 0.05*Seq1_bulk1 + … Each principal component has different weights, giving the final projection into three dimensions. The top 10 features that determine the three principal components calculated in AIMS, and their weights, are included in the output files "pca_compX_top10.dat". This has the added benefit that users can go back and determine which features are the most important for generating distinct clusters of receptors.

Regarding the determinism of PCA, for any given biophysical property matrix, i.e. any fixed input data, the exact same principal components will be identified. This is because the principal components are fundamental linear algebraic features of the matrix in question. For further reading into how PCA works and some of the mathematical fundamentals behind it, interested users can look into the general concept of spectral decomposition. For those really interested in a deep dive into linear algebra, David C. Lay's "Linear Algebra and Its Applications" was a particularly useful textbook to read through.

Moving from the fundamentals of PCA, we can see below the use of PCA in AIMS to analyze two different datasets:



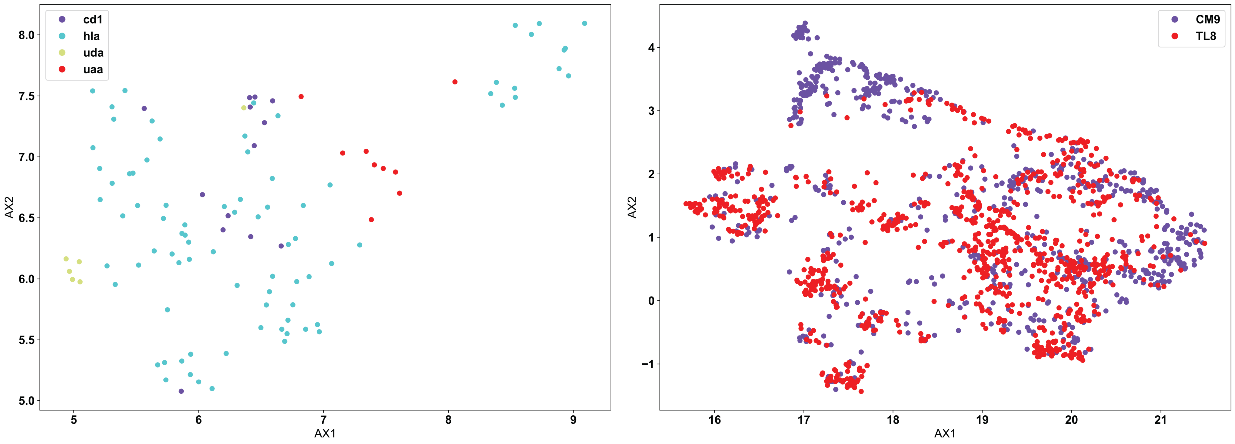
In line with what was discussed at the top of this page, it appears that PCA clusters the MHC data (left) quite well, but has a hard time of distinguishing a large portion of the TCR data (right). This is due to the nature of the input data. CD1 and HLA are two strongly biophysically distinct human molecules, whereas UDA and UAA are shark molecules. We see then that CD1 and HLA are distal in the PCA projection, while UDA and UAA are localized to the same region. Conversely, the TCRs in the right panel, specific to the CM9 and TL8 antigens of the Simian Immunodeficiency Virus (SIV) have a large number of biophysically similar receptors. This may be due to the nature of the data or could mean that another clustering algorithm is more appropriate in this instance.

UMAP

Over the past year or two, Uniform Manifold Approximation and Projection (UMAP, see more details here: https://umap-learn.readthedocs.io/en/latest/) has become increasingly popular in immunology as a means of projecting high-dimensional data onto a lower-dimensional space. Unlike PCA, UMAP is neither linear nor deterministic in nature. UMAP uses nonlinear, stochastic methods in an attempt to optimize the projection, with the overall goal of preserving the general structure of the sequences in the high-dimensional space. This means that no information regarding the "key components" or biophysical properties which determine the "closeness" of certain receptors can be gleaned from this projection. Further, it means that unless users set a specific "random_state" in their calling of the UMAP algorithm, there is no guarantee that their projection will be reproducible.

> **Warning:** In the AIMS GUI, the random_state is set to 47 for the UMAP projection, specifically so the results ARE reproducible. This is an introduced bias in the analysis, but guarantees that users can repeat their analysis consistently.

Using the same example input data as used above for the PCA example, we see below how the application of UMAP looks different:



Now, we see the reverse of what was seen for PCA. UMAP does not separate out the biophysically distinct MHC and MHC-like molecules (left), because UMAP is focusing more on the similar features of the molecules rather than the regions of highest variation in the data. MHC and MHC-like molecules are structurally very similar, and the differences are localized only in specific regions of the molecules. In this specific application, PCA is likely the more appropriate algorithm. Conversely, we can see in the TCR data (right) that there are still large regions of biophysically similar TCRs for each antigen specificity. Clearly, this is a fundamental feature of the input data here. We see multiple distinct small clusters of biophysically similar TCRs, compared to the three large clusters in the PCA analysis. Eventual clustering of these data will help to further break down the differences between PCA and UMAP.

t-SNE

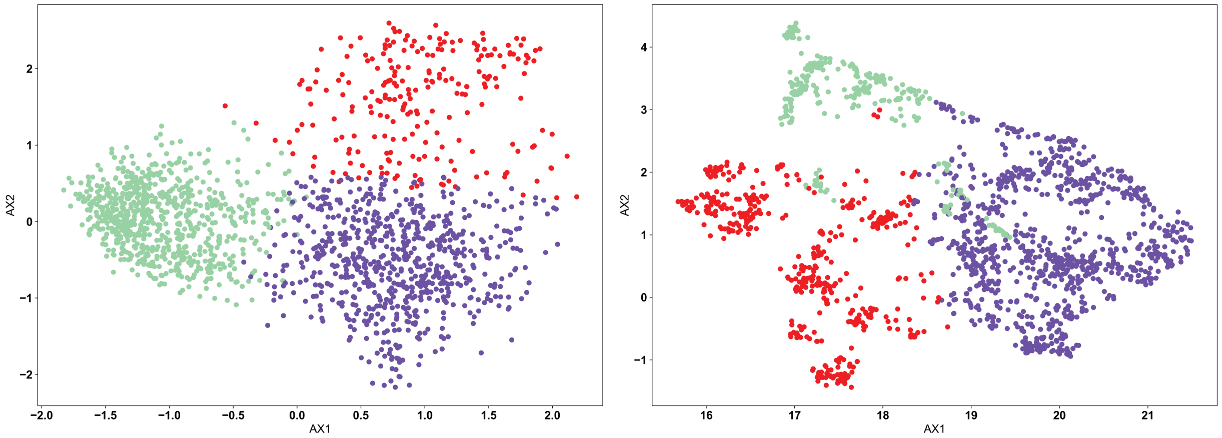
T-distributed Stochastic Neighbor Embedding (t-SNE) is largely similar to UMAP, in that is looking for a lower-dimensional projection of the data which preserves the general structure of the high-dimensional data. However, use of t-SNE is largely deprecated in AIMS, as it performs much worse than UMAP in all applications tested so far. t-SNE is no longer and option available in the AIMS GUI, but may still be used in the Jupyter notebooks. For more information on t-SNE and its use, see the SKLearn [the python implementation of t-SNE used in AIMS] documentation page (https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html).

1.4.2 Picking the Proper Clustering Algorithm

Once the dimensionality of the data has been properly reduced, we can often see by eye what appear to be distinct clusters of sequences. To be more quantitively rigorous, we turn to three specific algorithms to make these clustering decisions for us. Each of the clustering algorithms used in AIMS are distance-based, but their implementation varies in important ways. In some datasets, it may be obvious to the user how or why certain sequences cluster in certain ways. In the examples shown below, we will stick only with a non-obvious clustering example [the TCR data shown above], to show precisely how each algorithm performs in more extreme cases. For more information than is provided here, see the SKlearn clustering [the python clustering module used in AIMS] documentation (https://scikit-learn.org/stable/modules/clustering.html#clustering).

KMeans

The KMeans algorithm is perhaps the most conceptually simple of the three used in AIMS, and is the one most appropriate when there exists a strong reason a priori for a specific number of clusters. KMeans requires the user to pre-define the number of clusters, and then optimizes the number of points in each cluster based upon distance metrics. Importantly, how this optimization is done in the KMeans algorithm can cause "obvious" clusters to not be properly identified. The algorithm chooses centroids of each cluster, and attempts to minimize the distance from this centroid in determining the cluster identity. We can see this in the example below:

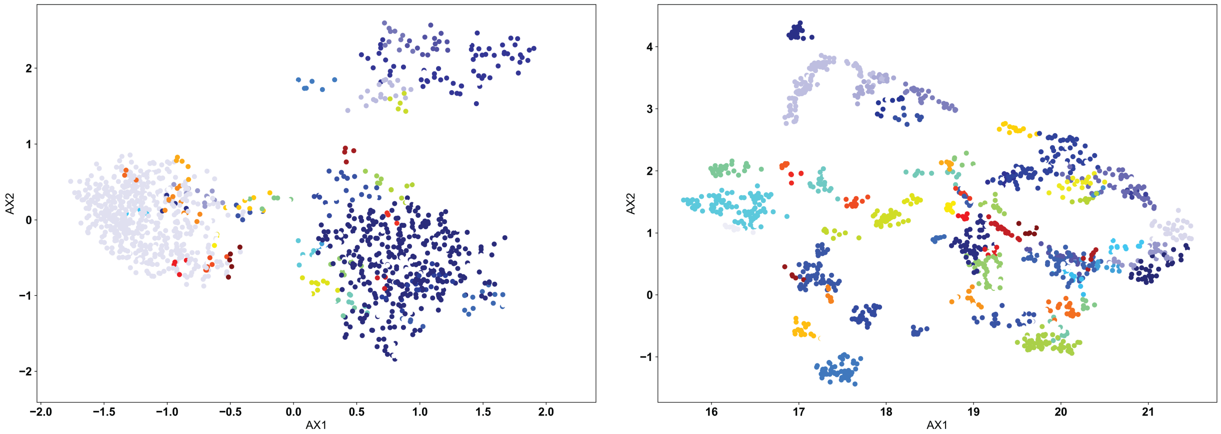


In each of the panels above, we can see that generally the three clusters are of the same size and shape, with variations in the point density. In the clustered PCA data (left, NClusts=3), we can see that there isn't any obvious reason why the data should be split that way, other than the fact that the algorithm was told to create 3 clusters. This is one of the dangers of using KMeans. The user can create arbitrary clusters of sequences that are not biophysically similar. The clustered UMAP data (right, NClusts=3) shows a similar erroneous clustering, whereby distinct clusters of sequences are included in a single Kmeans cluster, again because the algorithm was told it must search for exactly 3 clusters.

DBSCAN

In most AIMS applications, the "proper" number of clusters will not be obvious a priori, so either the OPTICS or DBSCAN algorithm should be used. The DBSCAN algorithm is a density based algorithm, identifying regions of high sequence density surrounded by regions of low sequence density. Due to the extremely high variance in the projected landscape of sequences, what constitutes a "proper" change in density must be user defined. In the AIMS GUI, users are responsible for setting a scan radius ('eps' in the AIMS notebooks, following the nomenclature of SKlearn). This radius tells the DBSCAN algorithm to look for clusters of points within a radius of this size. We can see how this looks with a scan radius of 0.2 below:

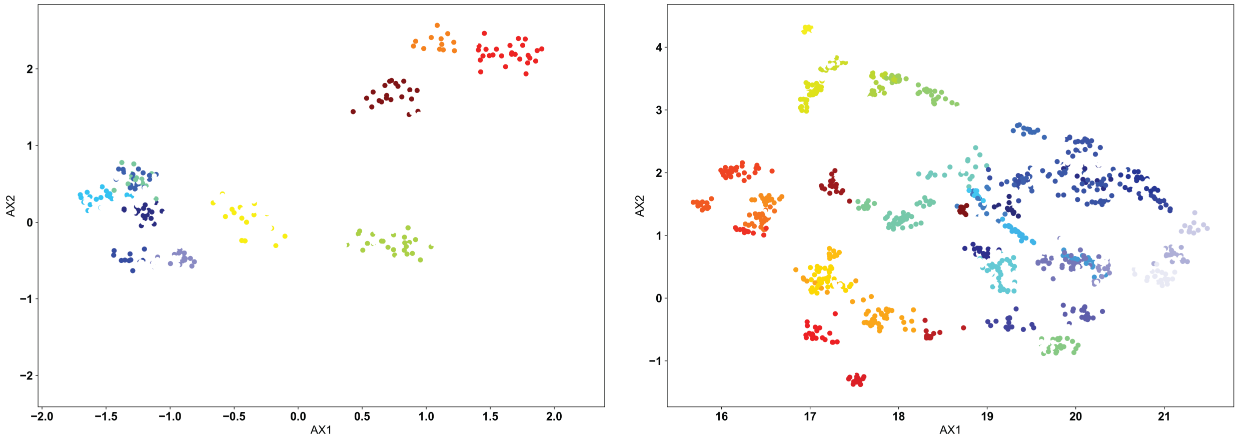
Right away, users should notice the difference compared to the KMean example: there are missing points in these figures. Both the DBSCAN and OPTICS clustering algorithms leave out certain points if they do not meet the criteria of what the algorithm considers a proper cluster. This is a key feature of these density-based algorithms, and may or may not be a desirable feature depending on the application. We can see in the clustered PCA data (left) that three large clusters are first identified, followed by a few "islands" of biophysically distinct clusters on the edges of these large



clusters. There are no such large clusters in the clustered UMAP data (right), and instead we see that the many distinct biophysically similar TCR clusters appear properly clustered.

OPTICS

The OPTICS algorithm is conceptually similar to the DBSCAN algorithm, but with a more user-friendly metric for what constitutes a proper cluster. In the OPTICS algorithm, the user must define the minimum number of sequences that are allowed to be considered a cluster, and the algorithm will go from there in defining clusters based on this minimum number. Each cluster is defined based upon some minimum distance between points, satisfying the minimum cluster number. We can see the example below, with a minimum cluster size set to 10:



Again, as in the DBSCAN-clustered data, we see many unclustered points in both figures, removing sequences that are not strongly similar to others. However, we can see how the lack of explicitly setting a cluster search radius can occasionally cause issues. In the clustered PCA data (left), we see that the large clusters identified in the DBSCAN clustering are left unclustered. Instead, more tightly clustered sequences are identified as distinct clusters. Only through user inspection can the appropriateness of these clusters be identified. Perhaps the DBSCAN clusters are too broad, or perhaps the OPTICS clusters are identifying point mutants of a single sequence, and are too restrictive. We do see however that the clustered UMAP data (right) is largely similar to the DBSCAN results, perhaps performing even better by breaking some larger clusters into smaller groups.

Conclusions

With that, you should be properly equipped to use the dimensionality reduction and clustering algorithms available in AIMS to find biophysically similar sequences within your datasets. This guide is meant to help the user, but a more comprehensive understanding should be the goal for all users. As such, more background reading than is available on this page is strongly recommended for long-term AIMS users.

1.5 AIMS Jupyter Notebooks

While the GUI is useful for those with little to no experience using python, unfortunately given the time commitment required to create a functional GUI the available features incorporated can lag behind significantly. For the most up-to-date analysis pipelines, fastest code, and maximal flexibility, it is recommended that the Jupyter Notebooks are used. I have tried to explain step by step within the notebooks how to run them. It may take a while, but reading through the markdowns and comments should give the user a good feel for what each code block is doing.

1.5.1 Notes on Notebooks

If using Anaconda-Navigator, you should just be able to launch Jupyter Lab from the main application. If using terminal, you can install Jupyter lab using:

```
conda install -c conda-forge jupyterlab
```

and start using the notebooks by navigating to the notebooks directory in terminal and typing in:

```
jupyter lab
```

1.5.2 Notebook Options

Further info on what each of these notebooks does is coming soon!

AIMS_notebook

AIMS_single

AIMS_mhc

AIMS_peptide

cleaner_bootstrap

manuscript_analysis

therapeutic_analysis

make_pretty_alignments