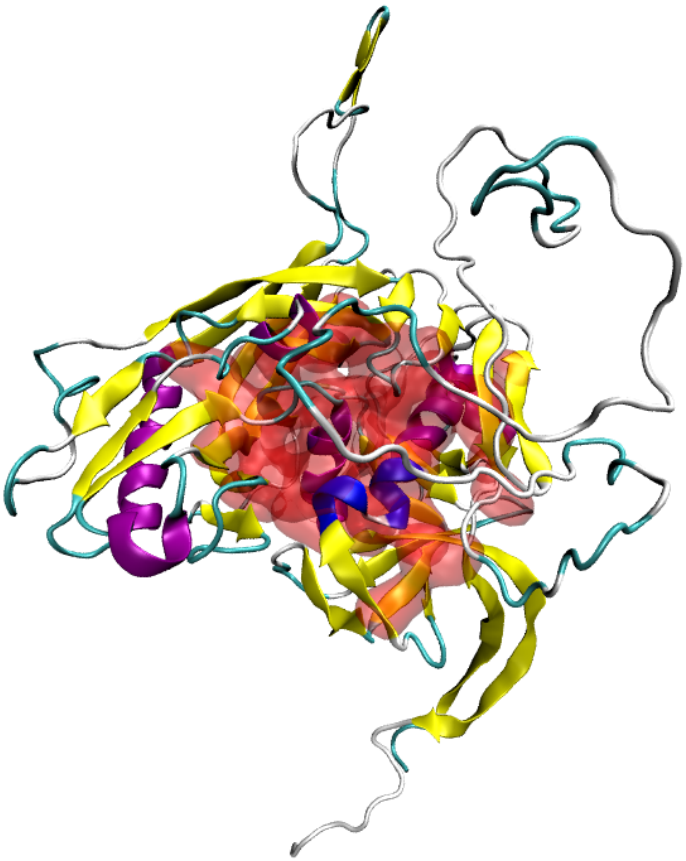
User Manual for ESSC, BESI, EVM

Scott P. Morton
Center for Computational Science
Middle Tennessee State University

January 20^{th}, 2019



Abstract

The process collectively is based on several programming languages interacting together to complete and provide the desired results. Through the use of Python, Cython, C, BASH, TCL, R and many direct system calls, the pipeline executes, to the extent possible, in an automated fashion to completion. The user should realize that, in order to execute, many pre- and post- execution processes will need to be completed. This guide will help the user prepare an execution environment and perform all steps needed to generate publish ready content.

NOTE - Please read through this document prior to making any adjustments, as some aspects will need decisions made before starting and an understanding of what is to be completed and when will ease the burden of setting this up for your specific use case.

Contents

1 Glossery of Terms

Abrieviated terminology is used in this document to ease typing requirements. Below is a list of terms and their shortened use in parenthesis.

- Configuration file (CF)
- Command Line Interface (CLI)
- Repository (repo)
- Human Immunodeficiency Virus (HIV)
- Message Passing Interface (MPI)
- Electro-Static Surface Charge (ESSC)
- Biomolecular Electrostatic indexing (BESI)
- Electrostatic Variance Masking (EVM)

2 Process Description (Brief)

As you read through this document, keep in mind that the primary use case for this software is the analysis of HIV and antibody protein structures, however, this does not mean that the pipeline could not be used for other purposes. Naming conventions are geared towards HIV and may be modified at some point to be more generalized as use cases are expanded.

2.1 ESSC

The pipeline performs each step to completion in the process to ensure that at any given point a plausible recovery can be made in the event of failure. The MPI based driver creates arrays of 64 bit integers (indexes) representing work units that are transmitted to each process. The index is then used to derive the work package just prior to execution. Failure of any individual process writes the index number out to log files that can later be used to re-index specific sections of the pipeline and executed to recover missed work. Figure 1 depicts the execution path of the pipeline.

2.2 BESI

The first analysis methodology is Principal Component Analysis (PCA) based and is derived from Latent Semantic Indexing (LSI). The method is synonymous with clustering based machine learning (ML) and is therefore ML-based in principal. We termed this process Biomolecular Electro-Static Indexing or BESI for simplicity of communication. The method performs a PCA of the surface charge data returned by the pipeline for all sequences involved. Using a known transmitted variant as the model for a successful transition of the transmission barrier as provided by Boeras et al., we make a Cosine Similarity Analysis of the first two principal components of each structure analyzed by averaging the principal component comparisons together. The process was evaluated across the three types of data produced by the pipeline, bound, unbound conformations and the difference of the two. The unbound conformational data is the only source that produced a significant signal to proceed with. This reduced the search space by two thirds and visualization of the process is represented by Table 1. BESI then generates a Phylogenetic tree and overlays the scores via a color gradient onto the tree leafs.

2.2.1 Phylogeny

The phylogenetic trees are inferred as follows: Sequences are aligned with MAFFT v7.273 using the L-INS-i strategy [1]. A maximum likelihood (ML) phylogenetic tree is inferred using the RAxML software, version 8.2.11 [2] with the HIVW amino acid model of substitution [3] and 100 bootstrap replicates. Trees are midpoint-rooted and rendered using APE version 5.0 [4]. Expression of the phylogenetic tree involves the use of color gradients in the vertical color bar that provides a reference to leaf objects and their BESI score where darker shades reference a closer fit to the control quasi-species and therefore predicts a higher chance of transmission.

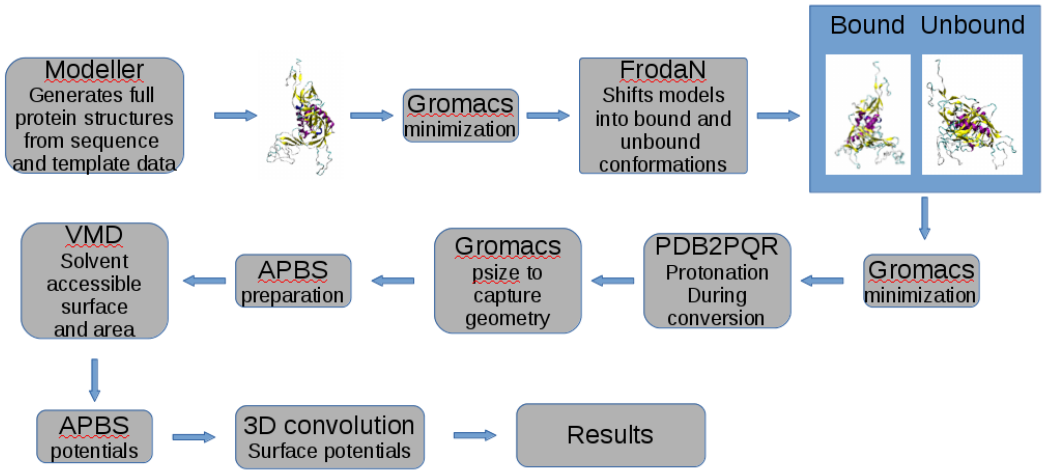


Figure 1: The Electro-Static Surface Charge (ESSC) pipeline execution path. Modeller creates protein structures that are energy minimized by Gromacs and perturbed by FrodaN into desired conformational states. After another minimization step, the structures are prepared in solvent titration of pH 3.0-9.0 in 0.1 increments and inserted into a box. The dimensions of the box are then determined and prepared for APBS. At this point the solvent accessible surface is determined and the structures atomic charges are calculated. Once charge calculation is complete, the surface charges are determined through a custom 3 dimensional convolution process. The resultant data is then processed through analysis methods describe in later sections.

Table 1: Visualization of the search space imposed by BESI. The process encompasses PCA of the sequence surface charge data, CSA comparison of the PCA data which is loosely based on latent semantic indexing.

Unbound Data						
Seq/pH	3.0	3.1	3.2	3.3	. . .	9.0
Seq1	0.008658	-1.246752	0.441558	1.229436	-0.060606	-1.290042
Seq2	0.017316	1.25541	-0.017316	0.580086	0.709956	-1.16883
. . .	0.017316	0.701298	-1.593072	-1.636362	-1.670994	-0.926406
SeqN	0	1.142856	-1.55844	1.549782	-0.597402	1.090908
Seq1 PCA Seq2 PCA . . . PCA 12 SeqN PCA		CSA	**BESI**	PCA is derived data while CSA is a comparison of the derived data.		

Table 2: Visualization of the search space imposed by EVM. This process encompasses the variance of all amino acids of all sequences processed during the study.

pH		Sequence Aligned Residues						
	Seq1	1	2	3	4	...	N	
3.0		0.017316	1.177488	-0.761904	0.770562	1.463202	1.445886	
3.1		0.017316	0.112554	-0.675324	-1.030302	-1.21212	-1.662336	
3.2		-0.008658	1.34199	0.30303	-0.380952	1.316016	0.883116	
...		-0.017316	1.549782	-0.926406	-0.320346	-1.463202	0.64935	
9.0		0.008658	-1.47186	0.380952	-0.441558	-1.108224	-0.536796	
	...							
	SeqN							
3.0		-0.017316	-1.350648	1.7316	-0.017316	-1.116882	-0.683982	
3.1		0.008658	-0.56277	-1.47186	1.367964	0.580086	1.177488	
3.2		0.017316	0.761904	-0.034632	-1.064934	0.90909	-1.55844	
...		0.017316	1.463202	-1.021644	-0.77922	0.926406	0.571428	
9.0		-0.008658	0.242424	-0.95238	0.233766	0.857142	0.493506	
		Variance	Hi-Variance	Variance	Hi-Variance	Hi-Variance	Variance	
	Electrostatic Variance Masking							

2.3 EVM

Selection of residues showing surface charge response to pH shifts involves calculating the electrostatic potential variance across all aligned sequences. Where gaps are encountered in the alignment, we utilize a value of zero (0). For each residue, the median value of individual residues for each model at a specific pH is taken to create a 1 x 61 vector for the pH range of 3.0 to 9.0 in 0.1 increments. The vectors are stacked row-by-row to create an array of dimension M x 61, where M is the number of sequences involved in the study. The mean value of each column is then calculated to produce a vector for which the variance is determined and stored. This is repeated for each alignment position. This method allows us to effectively filter out residues with small variations in mean surface charge across the pH shift. This technique uses residue numbers matching those of the HXB2CG standard naming convention as described by Korber et al. [5]. The search space imposed by EVM is visualized by Table 2.

3 Prerequisite Software

The following software will need to be installed prior to executing this pipeline:

- APBS
- PDB2PQR
- Modeller
- Gromacs - 5.x
- FrodaN
- Python2
- Python3
- MAFFT - 7.273
- RAxml - 8.2.11
- WebLogo - 3.6.0
- R
- OpenMPI
- tmux or screens

4 Directory Structures

The following directories can be created manual if you desire, however, to ensure consistent creation, a tool exists to create the folder structure upfront. By executing setup.py with the absolute path for the target study i.e. './setup.py /mnt/home/Studies/this-new-study'. Additionally, one could just copy the contents of 'Example' to the target via '-c' option of the setup utility. The following are root folders that need to be created and their expected contents:

- "ABseqfiles" - Contains anti-body or other secondary structure sequences files like CD4
- "ABtemplates" - Templates of the above for Modeller to build from
- "Analysis" - Analysis data from libBESI-EVM.r
- "seqfiles" - gp120 or primary structure sequence files
- "src" - This folder contains files specific to the individual third party packages like Modeller, Frodan, APBS, etc.
- "Structures" - This folder contains all created assemblies and data
- "targets" - This folder contains .PDB files for various aspects of the process for alignments and orientations
- "templates" - This folder contains the gp120 or other primary structure template files to be used by Modeller
- "tools" - Various scripts used to process components at certain points in the pipeline

The next list provides all the required folders for the analysis to be completed.

- "Analysis/Additional_Plots" - contains any additional plots such as EVM screeplot
- "Analysis/BE" - Contains binding energies plots and data
- "Analysis/BESI" - Contains BESI related plots
- "Analysis/Datasets" - All correlated data saved in various formats
- "Analysis/EVM_images" - EVM images
- "Analysis/FingerPrints" - Electrophoretic Fingerprints (Stieh et al.)
- "Analysis/HXB2" - HXB2 alignment data
- "Analysis/Logos" - Logos plot data
- "Analysis/PCA/FingerPrints" - We reverse the PCA data on the two principal components used and drop the graphs here to validate PCA processes.
- "Analysis/Residues" - Residue plots
- "Analysis/seqfiles" - sequence files
- "Analysis/VLoops" - Vloop plots and data
- "Analysis/Types" - Place your types file here

5 Preparations

In this section is described all the needed steps to complete before execution, please note that having the directory structure in place is assumed. Currently this system has dependencies on python2 because of certain requirements for FrodaN [6] and a couple of other third-party needs. The main driver and the pipeline are currently coded for python2 because of those dependencies.

It is therefore required to have: python2, numpy, cython, gcc and associated dev packages (numpy definately) installed and possibly others as this process has not been completely tested on a foreign system with other unknown dependencies missing. Please be patient and pay attention to errors you may encounter as those messages typically will guide you through to filling missing components.

Additionally, there are upcoming features that are being developed in python3. This can be quite frustrating at times as one learns to use the software, however, python2 simply will not support the requirements to move forward nor should it. As such, portion of this code will lag behind while updated versions of dependent software are developed. Specifically, FrodaN is one of those concerns as this package was developed for a Dissertation in 2010 and has not been updated since.

5.1 Compile ZFP

ZFP [7] is a high rate compression utility that can compact floating point arrays at orders of magnitude rather than rates of multiples. For the amount of data produced by APBS on a large scale execution, the space savings vs loss are completely acceptable and in smaller execution environments absolutely required. Highly recommend setting the compression flag in the CF and choosing an acceptable loss value to the highest possible.

Assuming you have extracted, cloned or used some other method to obtain a local copy of the repo, open a terminal session in the folder and cd into "libpyzfp/src" to execute:

```
./setup.py build_ext --inplace
```

This should create 'libpyzfp.so' in the current directory, copy this to the root of this repo folder for use on this specific environment.

WARNING If you upgrade, move or otherwise execute on a different platform, be sure to recompile and copy this library from the system environment for which you intend to execute. Not doing so will result in indeterminate behavior of the compression routine and most likely will result in unusable data from APBS [8].

5.2 Sequence Files

Copy the sequence files to be studied into the appropriate sub-folders created by using setup.py. Keep in mind that the primary structure files like gp120 are to be dropped into 'seqfiles' and secondary structures like CD4 into ABseqfiles.

5.3 Template Files

Modeller creates a likeness of the source sequence file, be it a gp120, CD4, Broadly Neutralizing Anti-Body, or any other structure that could be processed through the pipeline, by utilizing components fo the templates to 'fill in' the gaps in places such as variable loops in the case of gp120. As such, you need to have a variety of structures to draw from to ensure a level of randomness for completion of the models.

Modeller has a requirement to list the templates being used and therefore it is easier to name the files template1,template2, ..., templateN regardless of the type of sequence being created. The easiest solution is to create a soft link to the actual .PDB files being used. Copy the source template .PDB files into the appropriate folder and issue ln -s source.pdb template1.pdb for each of the templates you are providing.

5.4 Frodan

FrodaN [6] is a geometric targeting tool capable of perturbing structures into desired states with less computational time than traditional molecular dynamics. To avoid making potentially millions of copies

for files specific to Frodan, the pipeline sources a soft link from the src folder in the root study directory. samples of files required by Frodan are in the examples folder in repo for the pipeline. The following files must exist in the 'src' directory

- chains.txt - provides the chain ID's for gp120 or other primary sequence (always required)
- ABchains.txt - provides the chain ID's for BNAB's or other secondary structure complex structure and the CF is set to 1 for 'antibody' and 'multiChain'
- Complexchains.txt - provides chain alignment for the complex of a primary structure and multichain secondary structure and the CF is set to 1 for 'antibody' and 'multiChain'
- CD4chains.txt - provides the chain ID's for a single chain structure and the CF is set to 1 for 'antibody' and 0 for 'multiChain'
- options.xml - This is the Frodan CF as described in the FrodaN documentation found in the Docs folder of this repo

Chain files are simple configurations describing what the source and target assembly substructures are. The second line describes the target structure (reference). Below is a typical 'ABchains.txt'

```
A B
H L
```

Sample 'Complexchains.txt'

```
G H L
G H L
```

Do not attempt to add comments or any other additional spaces etc. into this file as your results will be indeterminate.

5.5 Types File

Preparing an analysis requires providing a 'types.txt' file in the 'Types' folder. Using a spreadsheet program like libreOffice calc or similar is easiest in most cases. Be sure your file has no header information when you are done, however, during creation the following column information might be handy to see at the top.

Table 3: Types file layout for R analysis script

Sequence Name	Clades	Subclasses	Donor/Recipient	Groupings

Clades and Subclasses are used for comparative plotting of structured data, for instance, the binding energies of Clade 'A' vs. Clade 'B' on a linear graph. Donor/Recipient (D/R) are used in phylogeny to distinguish BESI scores between subjects involved in the analysis of transmission amongst couples, while Groupings allow for distribution of a study that may involve multiple couples i.e. Boeras et al. A limitation for the column D/R is to use "D" and "R" as these characters determine the color scheme used for the tip label overlays.

To use this file in its simplest form, just list the sequence names of this study, the analysis will default all other values to a single type. On the other hand, if you only want to analyze Subclasses, you must put the sequence names along with a single value in the Clades column, make them all Clade 'A' for example. The same holds true along the list of columns, the program will not provide a default value for between columns, those defaults must be placed manually while the remaining empty columns to the right will continue to be filled with a default value.

You must be consistent in your use of values, for clades, subclasses and groupings. You can use any character(s) or number(s) so long as each is unique to the set of groupings. For instance, in the Clades column do not use 'A' and 'a' as identical definitions, use one or the other. Same holds true for subclasses and groupings where mixed values or letters can be used so long as group 'x' is the same set of characters consistently, otherwise additional groupings will arise in each of the columns. Consideration for the Groupings should be made as useful information, for instance, in the Boeras paper [9], couples were denoted by a common character set such as 'Z185' and is useful information for graphs of that couple. The tool will use this as part of the title in BESI graphs, and file names.

Table 4: Example types file content

R56FPL21apr05E7_plasmid_a	A1	TF	R	R56
R56FPL21apr05E7_plasmid_b	A1	TF	R	R56
R56MCA21aug0516_plasmid_9iii	A1	TF	D	R56
R56MCA21aug053_plasmid_5i	A1	TF	D	R56
Z153FPB13MAR02ENV1.1	C	TF	D	Z153
Z153FPB13MAR02ENV2.1	C	TF	D	Z153
Z153FPB13MAR02ENV3.1	C	TF	D	Z153
Z153MPB13MAR02ENV1.1	C	TF	R	Z153
Z153MPB13MAR02ENV2.1	C	TF	R	Z153

5.6 Sensitivity File

If you plan on tagging leaf nodes of the BESI phylo-tree to represent a pH sensitive variant, you will need to provide a sensitivity.txt in the 'Types' folder next to the required types.txt file. The format is simple, sequence name and TRUE or FALSE in a two column format. A value of TRUE will tag the branch with a red ●.

5.7 BESI Control Variant ESSC File

BESI utilizes a control variant that has been processed through ESSC with a number of models greater than or equal that being run for each study. The file contains electrostatic surface charge data for the target quasi-species and stored as a textual table in the Analysis/Datasets folder with the name "UB_mean_template.data." The name can be changed and adjusted in the driver for libBESI_EVM.r which is currently BESI_EVM_analysis.r.

5.8 Configuration File

The CF is JSON based for ease if use and broad acceptability across many platforms. Entries serve a specific purpose for the compute environment, or are study related and still others are general in nature. At one time there were plans for the ability to specify settings for individual sequences or a group (bulk_mode). Thus this setting is a leftover from those aspirations which may still be present in the code. It is suggested to store this json file in the root directory for this study to preserve the settings for future reference.

The following is a description of each Global setting:

- antibody - 0/1 antibody study no/yes respectively. Sets up for extended directory structure, calculations and related methods for complex structures
- ABmode - 0/1 antibody complex mode grid/mapped respectively where grid builds a complex structure from each Env to each BnAB ie Env1(model1) - BnAB1(model1), Env1(model1) - BnAB2(model1) ... Env1(modelN) - BnAB(X)(modelN) mapped mode reads a mapping from the CF as noted below
- mdrunThreads - The number of threads used by each subprocess for running 'mdrun' (MUST BE multiple of # logical cores per socket)
- OMPThreads - The number of threads to set for the OpenMP environment
- SeqThreads - The number of threads for simultaneous sequence processing (future/CNL specific use, must be a number)
- asyncThreads - The number of threads to allocate for single threaded processes or low intensity calculations like pdb2pqr (future/CNL specific use, must be a number)
- CompressData - 0/1 Use ZFP to compress DX file data, setting to 1 will compress, may get applied to other datasets in a later releases
- ZFPMaxError - 1e-1 thru 1e-9, scientific notation for acceptable error limit. At 1e-9 you will not gain much space and would probably do just as well to export the binary array

- tempDir - Directory name of a shared memory folder i.e. /dev/shm
- limitAPBS - Limit the APBS calculations to this number of models. Useful for data control. Performs all modeller and FrodaN operations at the 'models' setting. With this, a determination can be made of the need for all caclulations for APBS which produces a large data set. This is operated upon at the driver level
- multiChain - 0/1 Set multichain mode for minimization routines and complex structure procedures False/True respectively. Applies to secondary structures only ie BNAB, CD4 but not primary structures ie gp120
- socketsPerNode - Number of physical sockets (CPUs) on each compute node used to control mdrun core utilization
- physicalCores - Number of physical cores per socket used to control mdrun core utilization (cat /proc/cpuinfo — grep 'cpu cores')
- SMT-HT - AMD SMT (Simultaneous Multi-Thread) or Intel HT (Hyper-Threading) is enabled 1/2 (no/yes respectively) (core multiplier) used to control mdrun core utilization
- GMX-Margin - Determines the minimum distance gmx editconf should provide to clear the box edges when enclosing the structure. Try to keep this minimized to reduce computation time of the charges for each grid point of the box during APBS execution
- doRAnalysis - Execute R script to perform final analysis 0/1 (no/yes respectively). driver and library must be in root of study folder ('wdir' in next section of CF)

The following is a description of each Studies : bulk_mode setting:

- studyName - The name used for this study, will used on images, graphs etc. for titles etc.
- wdir - The absolute path of the working directory for this study
- models - The number of models desired from Modeller
- boundTarget - The bound target gp120 PDB file
- unboundTarget - The unbound target gp120 PDB file
- boundABTarget - The bound antibody target PDB file
- unboundABTarget - The unbound antibody target PDB file
- cgridsubsetFile - The pH-conc-grid file
- dockedTarget - The conformed complex target PDB file ie gp120 bound to BnAb or CD4
- gp120salign - The file name of the target gp120 coordinate file (PDB) used to pre-align this structure prior to merging with an AB or other secondary structure
- ABsalign - The file name of the target AB or other coordinate file (PDB) used to align this structure prior to merging with a gp120 assembly

```
{
    "Global":{
        "mdrunThreads":"4",
        "OMPThreads":"4",
        "asyncThreads":"4",
        "SeqThreads":"4",
        "CompressData":"1",
        "ZFPMaxError":"1e-1",
        "tempDir":"/dev/shm",
        "antibody":"1",
        "ABmode":"0",
        "limitAPBS":"10",
```

```
        "multiChain":"1",
        "socketsPerNode":"1",
        "physicalCores":"4",
        "SMT-HT":"2",
        "GMX-Margin":"2",
        "doRAnalysis":"0"
        },

    "Studies":{
        "bulk_mode":{
            "studyName":"Antibody Binding Energies",
            "wdir":"/home/scott/studies/AB-Study",
            "models":"3",
            "boundTarget":"1RZK-bound.pdb",
            "unboundTarget":"1RZK-unbound.pdb",
            "boundABTarget":"B12-bound.pdb",
            "unboundABTarget":"B12-unbound.pdb",
            "cgridsubsetFile":"pH-conc-grid-subset.txt",
            "dockedTarget":"3BNC117-docked_Target.pdb",
            "gp120salign":"3BNC117-docked_fit_prealign_G.pdb",
            "ABsalign":"3BNC117-docked_fit_prealign_LH.pdb"
        }
    }
}
```

5.9 Targets

For this part of the documentation we will be using VMD. there are alternates, this is just a familiar and common tool. Supposing that this is a start form scratch study, the following is a prerequisite.

5.9.1 GP120 or Primary Structure

A target gp120 protein must be prepared, for HIV the suggested assembly is 1RZK which currently is only found in the bound conformation. Using VMD, load the selected target and change the viewing mode by clicking on 'Graphics' of the main window and choose 'Representation.' Change the 'Drawing Method' to 'New Cartoon' and the 'Coloring Method' to 'Secondary Structures.' These selections will allow you to identify key components of the molecule for a desired orientation. For this example, we will manipulate the polymer to display the binding site with the alpha 2 helix and associated beta sheets to the left as shown in Figure 2.

For this part of the setup, orientation is less important for studies involving protein-protein interactions as the orientation of complex structures are used for the targeting of primary assemblies.

On the VMD Main window, add a new molecule using the menu items 'File' 'New Molecule.' From here there are two methods to manipulate the orientation to a desired view of the protein. Keeps these methods in mind as they will be used for setting up a complex structure later.

CLI Mode Click 'Extensions' and select 'Tk Console.' In the screen that opens type the following version 1.9.3 specific commands.

```
set sel [atomselect top all]
$sel move [trans x 15]
```

This will select all the atoms in this molecule and rotate around the x-axis by 15 degrees. Perform a similar action for each axis by some $+-$ value to situate the view as desired. Do not try to get an exact view for this exercise as we will fine tune the view in later steps. Next, click 'File' and 'Save Coordinates' to send this polymer to the targets folder of you study as 1RZK_bound.pdb or some other meaningful name.

Mouse Mode Click 'Mouse' then go to 'Move' and select 'Molecule' to enter move mode. A simple click of the mouse and you can drag the protein around to a desired position on the screen. Hold shift

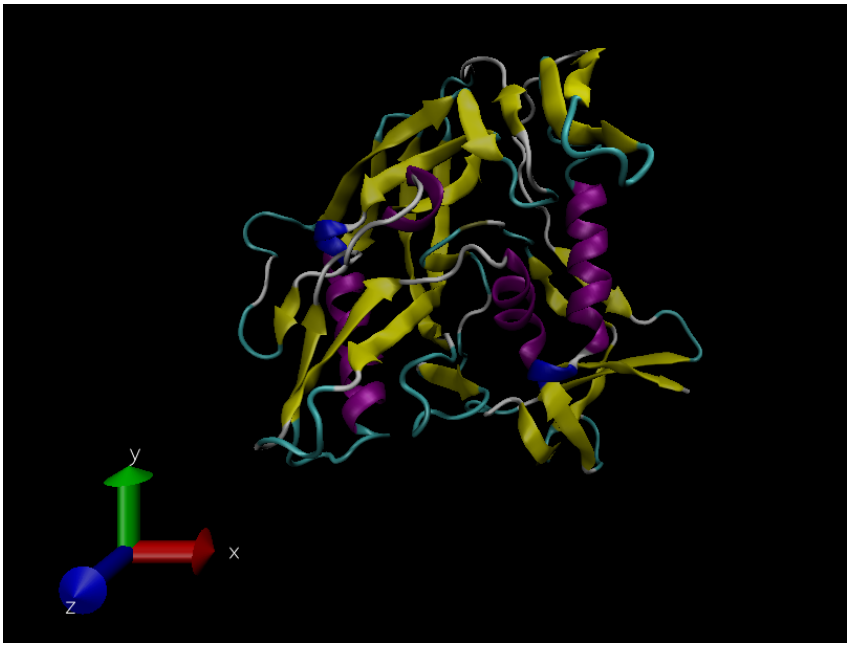


Figure 2: 1RZK oriented for use with the binding site shown forward and the alpha 2 helix to the left.

and click allows for rotation of the molecule. Again, click 'File' and 'Save Coordinates' to send this polymer to the targets folder of you study as 1RZK_bound.pdb or some other meaningful name.

I suggest using the 1RZK PDB files in the examples folder as this will save some time. However, if a pair of conformed structures are not available, utilize FrodaN to conform the assemblies to desired configurations. Save the files out to be used for "boundTarget" and "unboundTarget" in the CF.

5.9.2 Anti-Body or Other Secondary Structure

This section gets describe in what seems to be a bit backwards, we will start with the a view of the end state of the two proteins in complex with each other. In terms of gp120 and some antibody, this means that the gp120 has docked and bound to some substructure of the antibody protein. For this example, I will use 4JPV which is antibody 3BNC117 and gp120 93TH057 in complex.

What needs to be prepared is enough space between the two assemblies to allow variations of these structures and other such molecules to be graphed together with enough room between them to prevent interference when the two polymers are assembled together into a single container prior to merging them into the bound conformation. To keep it in perspective, Frodan will perform the final perturbation of the two assemblies into the bound conformation as we see in the original 4JPV pdb.

Open the structure in VMD and orientate the molecule in its entirety as previously described. In this example we have orientation to where the gp120 is on bottom and the antibody is on the top as shown in Figure 4. We will now break the assembly down into constituent components. On the VMD main menu select 'Graphics' and then 'Representations.' Next, click the 'Create Rep' button once. Then click the selections tab near the window center. Now click the 'reset' button. Under 'Keyword' select 'chain' and notice the list of chains in the 'Value' screen to the right. For this structure, the gp120 is chain 'G' and the antibody is chains 'H' and 'L.' For our purposes we will double click the word chain and then 'H' and 'L' so that the 'Selected Atoms' list shows 'chain H L' and click 'Apply. Next, click the first representation (or other depending on you curiosity) and click 'Reset' then find 'chain' again double click it and the 'G' chain so the 'Selected Atoms' are 'chain G' and click apply.

Next, select the representation of the structures for which we intend to move. In this example, I have selected chains H and L, clicked 'Mouse' in the main VMD window, opened 'Move' and selected 'Rep.' Over to the display window of VMD and drag the upper section of the presentation up to what appears as a reasonable amount of space between the two structures as shown in Figure 3

Alternatively, one could save the adjusted complex structure and the execute a Tcl script written to perform the same action. In the tools folder exists 'VMD_separate.tcl' created to perform the separation of two proteins where the gp120 and both chains of the antibody are moved to two times the distance of centers-of-mass.

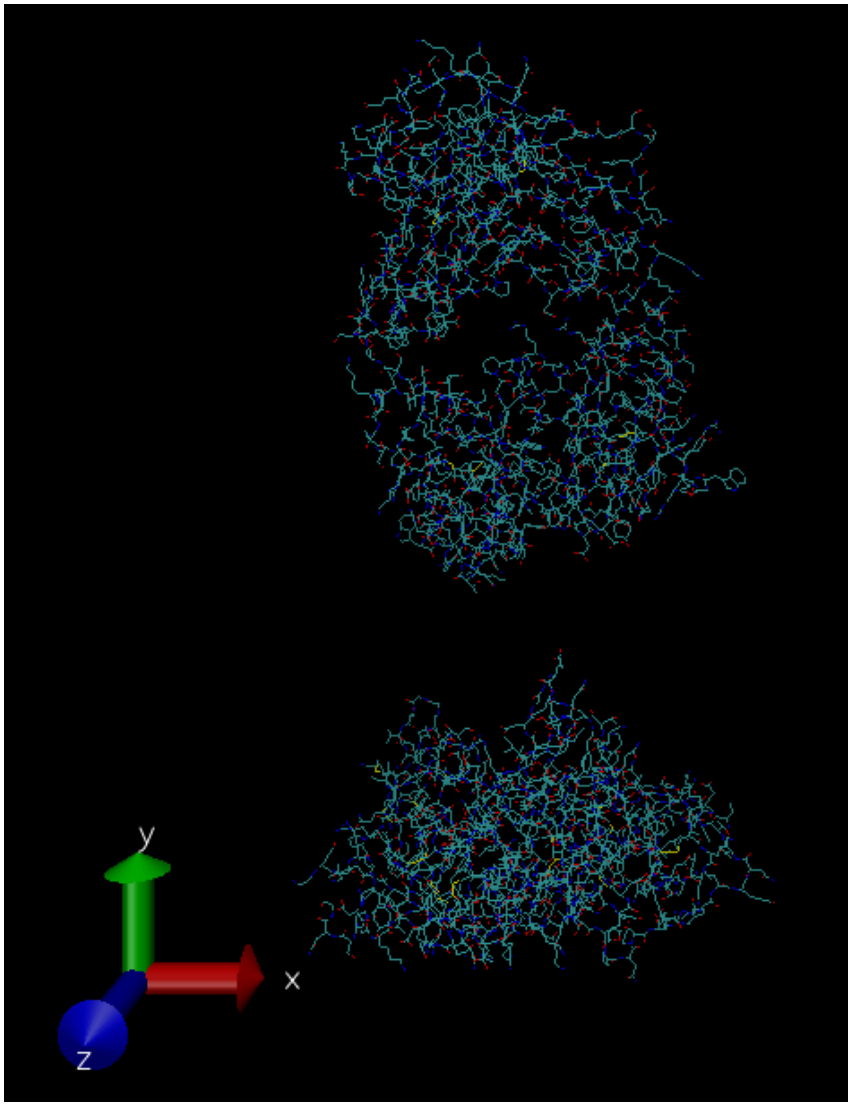


Figure 3: 4JPV separated in preparation of saving out to individual PDB files.

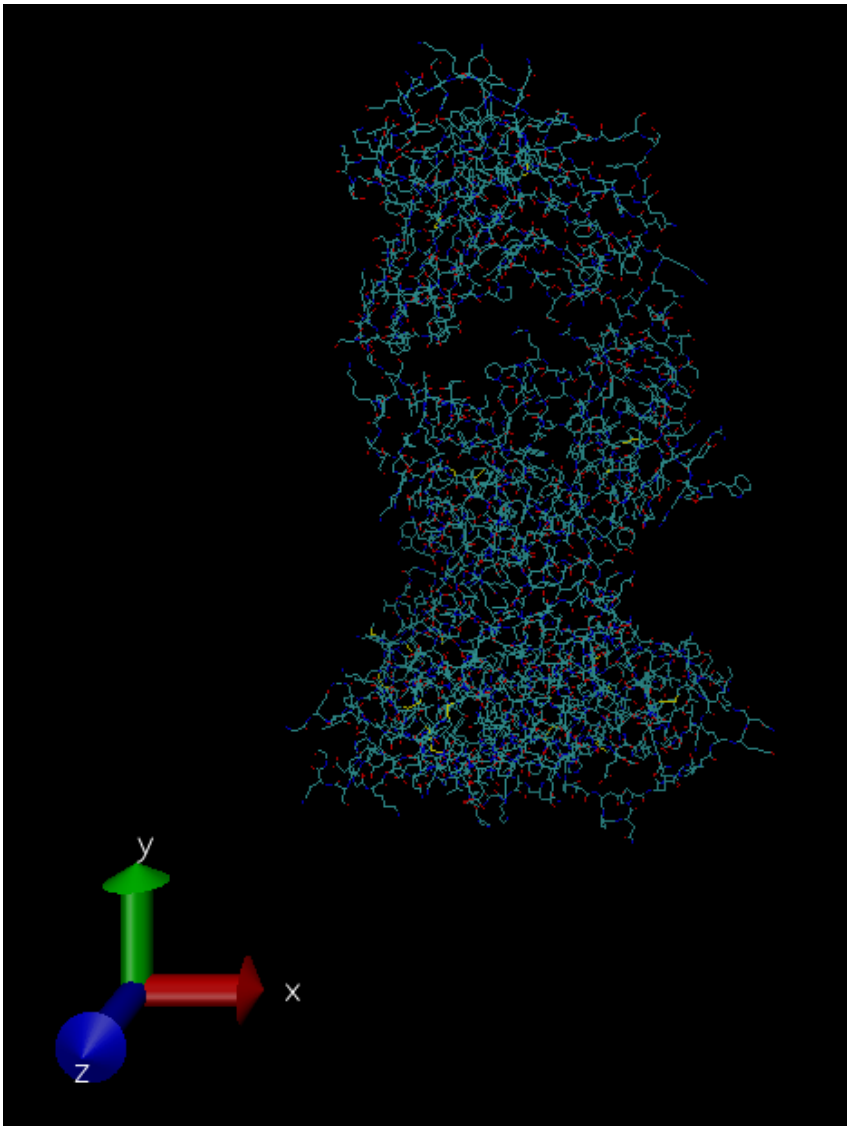


Figure 4: 3BNC117 in complex with a typical gp120 oriented for processing through the pipeline

```
vmd -dispdev none -e VMD_separate-V2.tcl -args 2ny7.pdb 0
```

at this point we are ready to save our separate structures as positioning targets for the process, click 'File' then 'Save Coordinates' in the main VMD window. Assuming this molecule is the only loaded structure, click the drop down for 'Selected atoms' and select 'chain G', click save and choose the target folder for this study and call this gp120SeparatedTarget.pdb and save it. Do the same for 'chain H L' and save this as ABSeparatedTarget.pdb and save this file.

Copy the source pdb to the target folder and set the name to be ComplexDockedTarget.pdb

These are the file names to use in the CF for "gp120salign", "ABsalign", "dockedTarget" respectively. Obviously, the file names can be changed to whatever is appropriate for the circumstance.

5.9.3 AntiBody

The simplest approach here is to find a structure in the Protein Data Bank (http://www.rcsb.org) [10] that is already an individual protein and use this structure for both the bound and unbound conformations of the AB molecule as no significant differences exist between the two. Subsequently, the previous section describes adequately the required steps to separate a complex pdb into individual components if needed. Save the files out to be used for "boundABTarget" and "unboundABTarget" in the CF.

You have now completed the required steps to prepare your protein structures for processing.

5.10 Analysis Configuration

The analysis is a set of Tcl, python, and R scripts that complete nearly every aspect of preparing print ready graphs and supplemental data for publishing. The setup requires a driver file to run the processing. The library file is libBESI_EVM.r and an example driver file is BESI_EVM_analysis.r both of which are in the Rscripts folder of this repo. Modify the driver file to fit your needs as this file has detailed comments to help you through the process. READ EVERY COMMENT to ensure you setup correctly.

5.10.1 EVM Tcl Script

This is a specific requirement for EVM imagery where Tcl does not allow for the passing of required data into the script via command line. In the tools folder you will find "VMD_gen_EVM_images.tcl" that performs all image creation for EVM. You will need to edit this file in a plain text editor. The script contains instructions on how to extract the rotation matrix from VMD, once obtained replace the current settings in this line and save the file:

```
set rmat "{{-0.673258 -0.738819 -0.0294399 0} {0.265488 -0.278708 0.922952 0} {-0.690098 0.613568 0.38379 0} {0 0 0 1}}"
```

For all intents and purposes, one should be able to open the target gp120 (bound or unbound), fine tune the view by simply rotating the object until a desired view is obtained, displaying the rotation matrix via

```
molinfo mol# get rotate_matrix
```

command, pasting into this script and saving. If no changes are made then the matrix should contain the following:

```
set rmat "{{1 0 0 0} {0 1 0 0} {0 0 1 0} {0 0 0 1}}"
```

It may take some trial and error getting this part right, we will go over the preliminary runs and fine tuning prior to a production run in a later section.

6 Pre-Production Execution

For this section we are going to perform a minor run of the setup to ensure everything works as expected and the results are produced as desired. We will limit our run to a single gp120 and a single secondary structure assuming we are performing a complex analysis. Depending on your desire to run BESI/EVM determines how to proceed for the test.

Fingerprints only, no BESI/EVM Drop a single gp120 and antibody sequence into the seqfiles and ABseqfiles folder of the study respectively. Change the number of models desired in the CF to 3. A suggestion would be to use those sequence files that are longest for this portion of the setup. As we will soon see, simulations can present other challenges over setting it up to run.

```
"models":"3",
```

Full BESI/EVM In order to infer a tree with RAxml you will need a minimum of four 4 gp120 sequence files in the seqfiles folder. All other considerations are the same as with Fingerprints only.

Review all other settings and make sure you have adjusted for the specific hardware that is to be the execution environment. Those settings in particular can be very detrimental to the success of any run.

If you are running 'module' then most likely the inclusion of pywrapper.bash is required to execute the system across multiple hosts. Edit this file to include the appropriate modules to execute the code. Several version of execution are listed below for various conditions that may exist:

- mpirun -n 6 pywrapper.bash pline.py ABS.jsn (mpich based run with all modules loaded via pywrapper and only one host. This is good for testing)
- mpirun –output-filename /misc/biosim3/spm3c/studies/AB-Study/rankoutput/rank.num -hostfile hostfile –mca btl_tcp_if_include bond0 –map-by socket pywrapper.bash pline.py ABS.jsn (openMPI based that logs all output to a folder, uses multiple hosts, has advanced networking enabled, and is mapped by socket in consideration of mdrun requirements to avoid processes stepping all over each other on high capacity nodes capable of running 20 or 30 processes. This is good for a production run)

Remember these are only examples, your execution environment will probably have very different requirements to run and all variations cannot possibly be provided.

7 Post Test Processing

Unfortunately, best coding practices indicate the need to fail hard and although the program will continue, there will come a time when processing just stops because an error is deemed too severe to continue.

The following is one of many potential errors that may be encountered and exhibits a prime example of being sure before making a production run. In this error PDB2PQR is complaining about missing atoms and this most likely due to the model extending beyond the box defined by Gromacs.

PDB2PQR Application Error

```
-------------------------
    PDB2PQR - a Python-based structural conversion utility
    -------------------------
    Please cite your use of PDB2PQR as:
    Dolinsky TJ, Nielsen JE, McCammon JA, Baker NA.
    PDB2PQR: an automated pipeline for the setup, execution,
    and analysis of Poisson-Boltzmann electrostatics calculations.
    Nucleic Acids Research 32 W665-W667 (2004).

    DEBUG INFO: PDBInputError /home/kyle/tmp/build/pdb2pqr/out00-PYZ.pyz/src.routines: 911
    Error encountered: Too few atoms present to reconstruct or cap residue ARG G 478 in structure!
    This error is generally caused by missing backbone atoms in this protein;
    you must use an external program to complete gaps in the protein backbone.
    Heavy atoms missing from ARG G 478: CB CA CG O N CZ NH1 NH2 CD NE
```

Opening up the related PDB file from the folder we garnished in the error logs reveals in fact that the structure did extend beyond the boundaries defined. Examine Figure 5 carefully in the lower left corner, we can see the atoms involved that are no longer attached to any protein in the structure. Gromacs' [11] triclinic box attempts encompass the assembly and typically performs well. Maybe we run a different model from the pool, or the orientation can be adjusted in a previous step (for all of them), or a more computationally drastic approach would be to adjust the box size to allow for this extreme variation.

But what is going on here? There actually two possible solutions to this question, the gp120/BNAB combination just makes a unique structure or protonation distorts the assembly to a point that portions of a protein extends beyond the defined box. Lets examine the logs as some detailed information can be garnished in this regards. One would expect that for the first hypothesis a gradual trend towards violation and the second would be all pH levels of a specific combination.

Grep the outputs for unique error message for this stage of the pipeline 'pdb2pqr failed in GenPsize for index:' by issuing the command:

```
cat * | grep 'pdb2pqr failed in GenPsize for index:' | wc
        61     427    2623
```

We can see from the output that 61 entries occurred, the exact number of pH values executed against per combination. Take another view of the logs

PDB2PQR Process Error

```
cat * | grep -B 11 'pdb2pqr failed in GenPsize for index:'

    GenPsize for All Structures for 1342
    Mon Nov  4 21:35:45 2019
    Current working dir is:
    /mnt/CNL/home/scott/studies/Test_1/Structures/Complex-B_6101_1_AY835434___3BNC117/apbs/bound/03/prot-0033
    Executing pdb2pqr (GenPsize)
    Command: "pdb2pqr.py --ff amber --ph-calc-method propka --with-ph=3 prot.pdb prot-3.pqr" completed

    pdb2pqr failed for : prot.pdb prot-3.pqr attempting to rerun process
    Executing pdb2pqr (GenPsize)
    Command: "pdb2pqr.py --ff amber --ph-calc-method propka --with-ph=3 prot.pdb prot-3.pqr" completed

    screamer - Driver error: pdb2pqr failed in GenPsize
    pdb2pqr failed in GenPsize for index: 1342
```

This sample output indicates the complex structure and model that failed, in this case all the failures took place on model 03. Potentially, the orientation may play a part in this problem, it is suggested that the assembly be adjusted to a vertical position prior to separation of components and subsequent

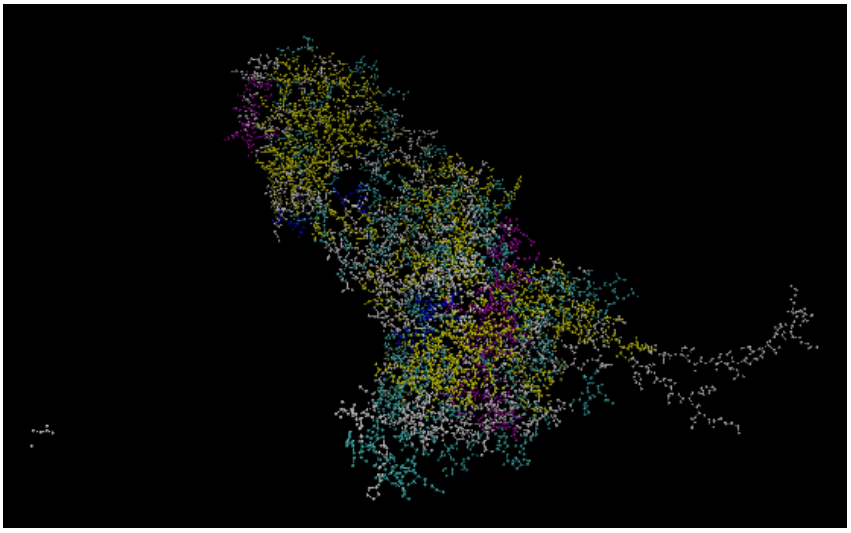


Figure 5: A complex structure that has violated the triclinic box defined by Gromacs. Observing the lower left corner displays a group of atoms that are detached from the gp120 protein in the complex structure.

execution of the pipeline. This is the orientation solution for all, if you still error out, then adjust the setting for GMX-Margin in the CF by +1 until the error is corrected. As indicated, this creates additional math for APBS to calculate and should be avoided to the extent possible.

For this particular run, reorientation of the complex structure corrected the issue. It is possible that a different model contributed, however, this particular combination of proteins was consistent about failing in the given parameters and was previously only correctable with a greater margin.

From here we take a look at some additional output data before proceeding. Open up the folder containing the study and navigate into the 'Structures' folder where all simulation work is stored. Pick you favorite structure and navigate into 'frodan/01/bound.' We can find several files ending in .gz where log details are stored in a gzip format to conserve space. Gzip files you see here are normal output and had there been an error, additional log files would be zipped for your examination. Take a look at this code snippet where pdb2gmx is being executed. Assuming the process fails, we write output to the console to be picked up by MPI's rankoutputs and a second attempt to process is performed, if this results in a failure the subprocess output is logged to a zip of the out put to a file name containing 'pdb2gmx' and the exception is raised to the driver resulting in this process failing out. This same logic is used throughout the pipeline and should assist in any failures one may encounter. Note the error messages above are taken directly from this information where pdb2pqr's application error is 7 and the process output is 7

```
cmdLine = 'gmx pdb2gmx -ignh -ff amber99sb-ildn -water tip3p -f %s -o conf.gro -ter -merge all' % (srcFile)
rvals = subProcessCMD(cmdLine,'Executing pdb2gmx for %s' % (srcFile))
if (rvals[2]):
    print 'pdb2gmx failed for %s attempting rerun' % (srcFile)
    print 'rc is: %s' % (rvals[2])
    print 'stderr has: %s' % (rvals[1])
    rvals = subProcessCMD(cmdLine,'Executing pdb2gmx for %s' % (srcFile))
    if (rvals[2]):
        print 'pdb2gmx failed for %s' % (srcFile)
        writeLogs(rvals,'pdb2gmx')
        raise Exception('Prep frodan - pdb2gmx failed to process')
```

Let us continue with the examination of output. Open the Analysis folder in the root of the study and look over some graphs in 'FingerPrints' to ensure our process is performing accordingly. Figure 6 shows the characteristic dip around pH 4.8 indicating correct processing of the pipeline for electrostatics. All fingerprint files are denoted with '_bound-unbound-total.mean' appended to the structure name. Lets take a look now at the EVM imagery output by opening up the folder 'EVM_images' in the root of this study. Opening up one of the '.pdf' files in this folder shows an EVM image that is not really oriented as we would like for this study as shown in Figure 7. As noted above, we will have to adjust the orientation and run the analysis again. Thankfully this is a small test run and this should only take a few minutes to complete.

Referring back to the Targets section 5.9 **for selection methods only**, open up one of the primary structures using VMD from bound or unbound in the frodan directory of this study and making the selections as previously describe. Press the 'r' key on the keyboard to ensure you have rotation selected

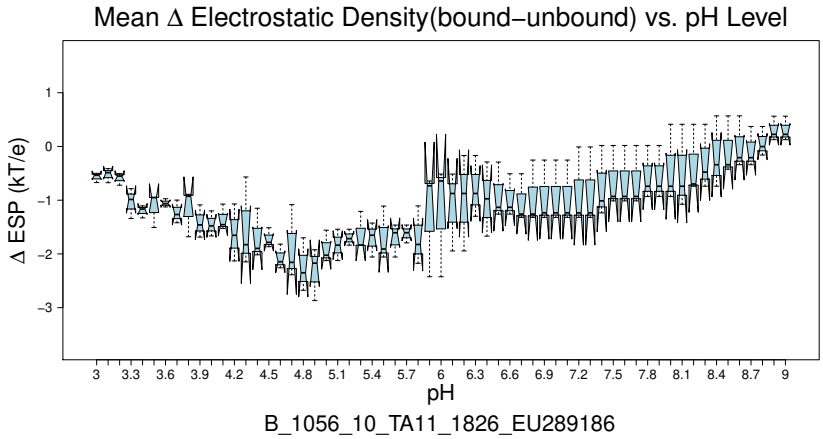


Figure 6: Typical electrostatic fingerprint showing characteristic signature with a dip around pH 4.8

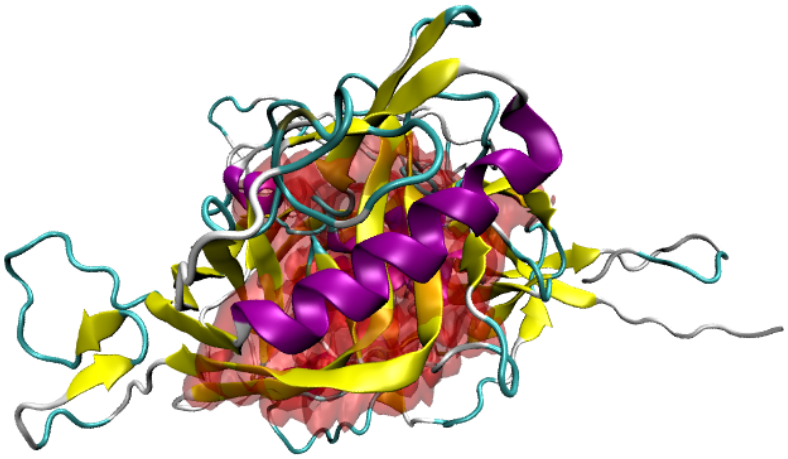


Figure 7: Bad orientation of EVM imagery

and rotate the assembly into a desired orientation. Capture the rotation as described in the EVM Tcl Script section 5.10.1 and save back out to the tools folder of this study. Execute the analysis by hand using the available methods described at the beginning of BESI_EVM_analysis.r to check that your results are indeed what is needed.

```
# Capture command line args and flag env for cli options
    # cliMode implies automation and assumes src and dest dirs are the same
    args = commandArgs(trailingOnly=TRUE)
    if (length(args) == 2){
        studyName <- args[1] # needs to be meaningful as this will be used in some graphics
        targetBase <- args[2]
        Dirs <- list(args[2])
    } else {
        # Adjust these 3 parameters for manual mode operation
        studyName = 'Binding Energies' # needs to be meaningful as this will be used in some graphics
        # Creates a subfolder here of variable "studyName"
        targetBase <- "/home/scott/mounts/biosim1/Studies/Test"
        # Dirs is a list to allow manual runs from multiple source folders into a single destination folder
        Dirs <- list("/home/scott/mounts/biosim1/Studies/Test")
```

8 Production Run

8.1 Hostfile

The pipeline is configured as a producer/consumer model and therefore needs an additional process to divvy out work to the consumers. Consider how MPI determines rank orders, the executing host is rank 0 and we need to consider this when we setup the hostfile that MPI will use to determine load on each compute node of the environment. Assuming you have hosts capable of running multiple instances of the process, and your execution node is host3, your hostfile may look like this:

```
host3 slots=17
host2 slots=16
host1 slots=16
host4 slots=16
```

and produces an execution environment of 64 consumers and 1 producer. The work load of rank 0 is small, however, this system is also responsible for executing the R analysis at the end of the pipeline as most head systems usually have more ram and cpu available.

From here, complete the setup by making any adjustments in the post test processing section, dropping the remaining sequence files in the appropriate directories and executing a production run command line considerate of the environment.

9 Advance Usage

Here we will talk about some special use cases of the pipeline along with BESI/EVM drivers.

Pipeline Assuming you have a specific use case or need to limit the execution pipeline to certain functions, one may open 'pline.py' and observe the 'messages' list created at or around line 197. Keeping in mind that Python is 0 indexed, a list or range of activities that need to be performed.

```
    messages = ['Preparing to build directories for Complex Structures',\
'Preparing to build directory structures and run Modeller',\
'Setting up to prep FrodaN',\
'Setting up to run FrodaN',\
Setting up to run PrepComplex',\
Setting up to run DockComplex',\
Setting up to run MergeALLPDB',\
Setting up to run GenPsize',\
'Setting up to run genComplexIN',\
'Setting up to run genIN',\
'Setting up to run GenSAS processes',\
'Setting up to run APBS processes',\
'Setting up to calculate means and ratios for all individual structures',\
```

```
'Setting up to calculate Binding Energies for Complex structures',\
'Setting up to calculate Residue surface charges',\
'Setting up to correlate Residue surface charge data']
```

Now move down the code a little at or about line 228, where 'Stages' is defined and 'workType' a few lines later. One could use the commented 'workType' entry to define a list of activities or even a non contiguous list through one of the following methods:

```
    Stages = [structuresWork,modellerWork,
prepFrodaNWork,runFrodaNWork,
ComplexWork,ComplexWork,
ComplexWork,APBSWork,
ComplexgenINWork,APBSWork,APBSWork,APBSWorkLTD,
resultsWork,structuresWork,
ResWork,resMnRWork]

workType =  range(0,len(Stages))
#workType =  range(13,14) # For stepping through
#workType = [0,4,7,9]
```

Keeping in mind that the pipeline has dependencies on previous steps , these are just examples of how the engine can be utilized. Additionally, one may need to run specific subsets of a certain work type. if we look just before where 'Stages' is defined, we see the work type are just lists of ranges, in our error above, a range of indexes were in error, we could change the definition of a work type to be that list of failed indices, adjust the 'workType' list to perform just that work type and execute on those specific failures only. This could be done over and over again until a solution is found without re-executing all the previous work that was completed.

This demonstrates the flexibility and power of the engine that runs the pipeline.

BESI/EVM - Inspecting the R script 'BESI_EVM_analysis.r' allows the user to determine specific use cases for this driver as well. Read through the comments to easily determine what portions of the script needs to run or rerun if that is the case. There are control statements at or about line 56 that control subsets of execution with TRUE/FALSE statements. Keep in mind that this file is all about controlling output of the library, look this file over carefully before pre-execution and adjust as needed when production execution is ready to take place.

You can adjust parameters at or about line 35 to execute in manual mode to specific locations outside of the studies root folder. The directory/file structures have to be in place with all requirements listed above for this portion of the pipeline.

References

[1] Kazutaka Katoh and Daron M. Standley. MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Molecular Biology and Evolution*, 30(4):772–780, 2013.

[2] Alexandros Stamatakis. RAxML version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313, 2014.

[3] David C. Nickle, Laura Heath, Mark A. Jensen, Peter B. Gilbert, James I. Mullins, and Sergei L. Kosakovsky Pond. HIV-specific probabilistic models of protein evolution. *PLoS ONE*, 2(6), 2007.

[4] Emmanuel Paradis, Julien Claude, and Korbinian Strimmer. APE: Analyses of phylogenetics and evolution in R language. *Bioinformatics*, 20(2):289–290, 2004.

[5] B Korber-Irrgang, BT Foley, C Kuiken, SK Pillai, JG Sodroski, JG Sodroski, B Foley, T Leitner, S.K. Pillai, and J.G. Sodroski. Numbering positions in HIV relative to HXB2CG, jan 1998.

[6] Daniel W. Farrell, Kirill Speranskiy, and M. F. Thorpe. Generating stereochemically acceptable protein pathways. *Proteins: Structure, Function and Bioinformatics*, 78(14):2908–2921, 2010.

[7] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Trans. on Visualzation and Computer Graphics*, 20(12):2674–2683, 2014.

[8] N A Baker, D Sept, S Joseph, M J Holst, and J A McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proceedings of the National Academy of Sciences of the United States of America*, 98(18):10037–41, 2001.

[9] D. I. Boeras, P. T. Hraber, M. Hurlston, T. Evans-Strickfaden, T. Bhattacharya, E. E. Giorgi, J. Mulenga, E. Karita, B. T. Korber, S. Allen, C. E. Hart, C. a. Derdeyn, and E. Hunter. Role of donor genital tract HIV-1 diversity in the transmission bottleneck. *Proceedings of the National Academy of Sciences*, 108(46):E1156–E1163, 2011.

[10] H. M. Berman. The Protein Data Bank . *Nucleic Acids Research*, 28(1):235–242, jan 2000.

[11] H.J.C. Berendsen, D. van der Spoel, and R. van Drunen. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications*, 91(1-3):43–56, sep 1995.

10 Conclusion

After browsing through the graphs and imagery produced one should be able to recognize that most of the work is complete and only a few minor adjustments like cropping images down to fit the manuscript are required assuming time was taken to complete the setup as described in the user guide.