Table of Contents

[Introduction to ioNERDSS 1](#_Toc157896751)

[Description 1](#_Toc374750920)

[Section Descriptions 1](#_Toc1308596814)

[Function Formatting 1](#_Toc1942980868)

[Creating NERDSS Inputs 2](#_Toc2145332038)

[Database PDB 2](#_Toc61140539)

[real\_PDB\_UI () 2](#_Toc1602983)

[MAIN OBJECT - Read in PDB File to create protein complex object (PLEASE READ!!) 2](#_Toc1184852540)

[MODIFY - Filter the protein complex object, so it only includes the chains you want 3](#_Toc722270919)

[MODIFY - Change the distance between 2 binding sites 3](#_Toc1184632477)

[MODIFY - Calculate the angles between pairs of interfaces 3](#_Toc1412930218)

[MODIFY - Normalize the COM of each chain 3](#_Toc1698258485)

[OUTPUT - Writes new NERDSS input files based on chain info 3](#_Toc818222495)

[OUTPUT - Writes a new PDB file based on chain info 3](#_Toc909329160)

[OUTPUT - Create a 3D plot of each inputted chain 4](#_Toc1667602303)

[Platonic Solid Self-assembly Input File Writing 4](#_Toc201423459)

[Analyzing NERDSS Outputs 4](#_Toc736426486)

[Analyzing Histogram Files (General Functions) 4](#_Toc1066650141)

[MAIN OBJECT – The Main Single-component Histogram Object (PLEASE READ!!) 4](#_Toc1234025335)

[MAIN OBJECT – The Main Multi-Component Histogram Object (PLEASE READ!!) 4](#_Toc2098858770)

[DATAFRAME – stores count of each complex type for each timestep 5](#_Toc233521408)

[CSV – stores count of each complex type for each timestep 5](#_Toc10889770)

[LINE GRAPH – max count of protein species in a single complex at a time 5](#_Toc668717203)

[LINE GRAPH – mean count of protein species in a single complex at a time 5](#_Toc1100330286)

[Analyzing Single-component Histogram Files 6](#_Toc1466339199)

[HISTOGRAM – average number of each complex species size: 6](#_Toc1851868324)

[3D HISTOGRAM – relative occurrence of each species over time: 6](#_Toc1910377748)

[HEATMAP – average number of complexes at each time interval: 7](#_Toc379052431)

[HEATMAP - total count of monomers in each complex size vs. time: 7](#_Toc165066961)

[HEATMAP - fractions of original monomers in each complex species vs. time: 7](#_Toc2079225343)

[Analyzing Multi-component Histogram Files 8](#_Toc1209256977)

[HISTOGRAM – Frequency of each complex size 8](#_Toc1282397730)

[STACKED HISTOGRAM – Counts of complex species with certain protein compositions 8](#_Toc259541732)

[HEATMAP – Average count of each complex composition over simulation time 9](#_Toc951116115)

[3D HISTOGRAM - Average count of each complex composition over simulation time 9](#_Toc335065891)

[Analyzing Transition Matrix Files 10](#_Toc572830219)

[\*Incomplete LINE PLOT - free energy among different size of complexes: 10](#_Toc1084092991)

[\* Incomplete LINE PLOT - symmetric associate probability: 10](#_Toc1710009287)

[\* Incomplete LINE PLOT - asymmetric associate probability: 10](#_Toc213186628)

[\* Incomplete LINE PLOT - symmetric dissociate probability: 11](#_Toc1167899362)

[\* Incomplete LINE PLOT - asymmetric dissociate probability: 11](#_Toc495435644)

[\* Incomplete LINE PLOT – Growth probability for each complex size: 11](#_Toc1979627603)

[LINE PLOT – Average lifetime for each complex type: 11](#_Toc2050419190)

[Locating position for certain size of complexes by PDB/restart file 12](#_Toc1276328575)

[By PDB file: 12](#_Toc1360741575)

[By ‘restart.dat’ file: 12](#_Toc1090192286)

[Analyzing .xyz files 13](#_Toc316163165)

[CSV – creates spreadsheet of protein locations 13](#_Toc1043253264)

[DATAFRAME – creates dataframe of protein locations 13](#_Toc1906567190)

[MATRIX - tracks the trajectory of specific protein(s) 13](#_Toc1730331994)

# Introduction to ioNERDSS

### Description

ioNERDSS is a python library to create Inputs and analyze outputs from the NERDSS simulator software. Intended to be reasonably easy to use, and fast.

### Section Descriptions

This user guide is separated into 2 main sections, each with a couple sub-sections.

**Creating NERDSS Inputs**: Automatically creates NERDSS inputs for you!

* Database PDB: This will create NERDSS inputs based on a PDB file downloaded from the databank. All functions are methods of the ProteinComplex object, except for UI.
* Platonic Solid: This will create NERDSS input based on the type of solid it will create when fully assembled. There are 2 options for each of the 5 platonic solids.

**Analyzing NERDSS Outputs:** Creates graphs, spreadsheets, and more from NERDSS outputs

* Histogram Section: As multi/single species histograms are so similar they share many attributes. All histogram functions are methods of the 2 main objects listed in General.
  + General: Initializing the main objects, and methods shared across both object types.
  + Single Species Histograms: Methods specifically for the single species object.
  + Multi Species Histograms: Same as above but for 2+ species (2+ protein types)
* Complex Location: This will read in PDB + restart or input files to determine location of certain complex sizes
* Transition Matrix: This will read in the transition matrix file and create a variety of outputs
* XYZ: This will read in .xyz files which hold the location for specific timestamps, and create a variety of outputs

### Function Formatting

Each sub-section will have a variety of useful functions listed. This is what each section means

\*NOTE: Each section that has [] around it means that it will be replaced by something in the actual description. These words inside of the brackets describe what that will be.

\*\*NOTE: Each section that is *italicized* means it will not be included in every function

Function Format:

[function-name] ([function parameters])  
Desc: [short description]

Parameters:

* [parameter-name] ([parameter type*], [if-optional] = [value-if-unset]):* [Description]
* *This will be repeated for each parameter*

*Long Description: [this will be a longer description if necessary]*

Example: [shows example of function being used, possibly with images]

# Creating NERDSS Inputs

This section is based around automatically creating inputs for NERDSS through a variety of methods such as:

## Database PDB

These functions will use .PDB files downloaded from a protein databank to create a NERDSS input.

### real\_PDB\_UI ()

Description: This function will read in the PDB file and create a NERDSS input with a user-friendly UI. Can not be run in a Jupyter Notebook.

\* There are no inputs in the original function, you just need to write **real\_PDB\_UI()** into command line or IDE.

**Tutorial:**

First, store a real PDB file in the same path with the python code and call this function with an appropriate IDE (here take VSCode for example). The interface will require the user to input the name of the desired PDB file. Type in the full name of the file and press return to continue. See screenshot below for details.



Once the file name is input, the code will read the desired information inside this PDB file and show some basic parameters on the interface (this will take a while), including the name of each chain, size of each chain, the coordinate of each COM and each pair of interfaces.

A picture containing table

Description automatically generated

Among all pairs of interfaces, you are asked if you want to change the distance between interfaces (AKA sigma), if you enter ‘yes’, you can change any of the distance shown above or change all distances to the same value; if you enter ‘no’, the distances will not be changed.

Graphical user interface, text

Description automatically generated with medium confidence

Then, you are asked if you want to use the default vector (0,0,1) as the normal vector. If you type in ‘yes’, normal vectors for all interfaces will be set as (0,0,1); If you type in ‘no’, users are able to manually input desired normal vectors in a format of ‘1,1,1’ (without parentheses). If a colinear issue takes place, the algorithm will automatically detect it. If the default vector was used, it will use (0,1,0) instead, and for manually input vector, you will need to input a new vector.

Text

Description automatically generated

At last, you are asked if you want each chain to be centered at COM. If you write ‘yes’, the COM coordinate will be normalized as (0,0,0) and the corresponding coordinates of all interfaced will be all changed accordingly in the final output; if you write ‘no’, the coordinates for all COM and interfaces will stay the same as the original ones.



The code will then automatically quit and the corresponding input (multiple .mol files and single .inp file) will be found in the same directory as the Python file.

**IMPORTANT:** All future functions are methods of the Protein Complex object (read is the \_\_init\_\_ function)! They can be used separately as well, where you set the Output of each function to the result variable, and add an ‘result = result’ parameter, however it frequently breaks.

### MAIN OBJECT - Read in PDB File to create protein complex object (PLEASE READ!!)

real\_PDB\_separate\_read (FileName,ChainsIncluded)

Description: Reads in a database PDB file and finds information important for NERDSS inputs.

Parameters:

* FileName (String): The full path of the desired PDB file or name of the file if in same directory.
* ChainsIncluded: (list, optional)  
  Description: A list of which chains you want to be included. Must be more than 2.

Example:

Clathrin = ioNERDSS.ProteinComplex(FileName=”ioNERDSS\Test\database.pdb”, ChainsIncluded=['A’,’ B’,’ P’,’ Q’]

>>> *Creates a Protein Complex object called Clathrin that holds the data in the database file*

Long Description: This function will extract the coordinate information stored inside a real PDB file and calculate the COM of each unique chain, as well as recognize the binding information between each pair of chains (all atoms of different unique chains that are closer that 3.0 angstroms are considered as bonded), including whether two chains are bonded and the coordinates of each binding interface. All the information will be printed on the screen and the returns will contain all the information for further analysis.

### MODIFY - Filter the protein complex object, so it only includes the chains you want

dtb\_PDB\_filter(ChainList)

Description: This function will filter the desired chain according to the input list of chains.

Parameters:

* ChainList (List with String elements): The desired name of chains that users intend to examine.

Example:

Clathrin.dtb\_PDB\_filter(ChainsIncluded=['A’,’ B’,])

>>> *Edits clathrin so it only included the A and B chain*

### MODIFY - Change the distance between 2 binding sites

dtb\_PDB\_change\_sigma (ChangeSigma, SiteList, NewSigma)

Description: Changes the value of sigma, the distance between two binding interfaces.

Parameters:

* ChangeSigma (Bool, optional = False): Whether the sigma values will change or stay the same
* SiteList (List with Int elements, optional): It consists of the serial numbers of the pair of interfaces for which the user needs to modify the sigma value. The serial number is determined by the pairing sequence shown by the function ‘real\_PDB\_separate\_read’. If the serial number is 0, it means to change all pairs of interfaces into the same sigma value.
* NewSigma (List with Float elements, optional): It consists of the actual sigma value that users desire to change, according to the sequence of input ‘SiteList’.

Example:

Clathrin.dtb\_PDB\_change\_sigma(ChangeSigma = True, SiteList = [1,2], NewSigma: [1.01])

>>> *Chanes distance between first and second chain to 1.01 angstroms*

Long Description: This function allows users to change the value of sigma (the distance between two binding interfaces). The new sigma value and the corresponding coordinates of interfaces will be shown on the screen and the returns will contain all the information for further analysis.

### MODIFY - Calculate the angles between pairs of interfaces

dtb\_PDB\_calc\_angle ()

Description: Calculates the 5 associating angles of each pair of interfaces. After this runs, it is ready to be output

Parameters:

* None

Example:

Clathrin.dtb\_PDB\_calc\_angle()

>>> *Finds the 5 associating angles for each interface. Ready to be output now.*

Long Description: Calculates the 5 associating angles of each pair of interfaces. The default normal vector will be assigned as (0, 0, 1). If the co-linear issue occurs, the system will use (0, 1, 0) instead to resolve co-linear issue. The calculated 5 angles will be shown on the screen automatically. If user intends to manually input the normal vector, please refer to function ‘real\_PDB\_UI’, the separated function does not support manual inputs. The returns will contain all the information for further analysis.

### MODIFY - Normalize the COM of each chain

dtb\_PDB\_norm\_COM ()

Description: Normalizes the COM of each chain as (0, 0, 0). The interface of each chain will be subtracted by the COM coordinates accordingly.

Parameters:

* None

Example:

Clathrin.dtb\_PDB\_norm\_COM()

>>> *Center of Mass and all binding locations are now set around 0,0,0.*

Long Description: Normalizes the COM of each chain as (0, 0, 0). The interface of each chain will be subtracted by the COM coordinates accordingly. Once the calculation is completed, there will be a message shown on the screen. The returns will contain all the information for further analysis.

### OUTPUT - Writes new NERDSS input files based on chain info

dtb\_PDB\_write\_input ()

Description: This function will write ‘.inp’ and ‘.mol’ files according to all the calculations and modifications above.

Parameters:

* None

Example:

Clathrin.dtb\_PDB\_write\_input()

>>> *Creates new .inp and .mol files ready to be input into NERDSS*

Long Description: This function will write ‘.inp’ and ‘.mol’ files according to all the calculations and modifications above. Multiple ‘.mol’ file and a ‘.inp’ file can be found in the same directory as the Jupyter Notebook file once the function finish running.

### OUTPUT - Writes a new PDB file based on chain info

dtb\_PDB\_write\_PDB()

Description: This function will generate a PDB file that only contains the calculated COMs and reaction interfaces for visualization and comparison with the original PDB file.

Parameters:

* None

Example:

Clathrin.dtb\_PDB\_write\_PDB()

>>> *Creates new .pdb files to be compared with original inputted .pdb file*

Long Description: This function will generate a PDB file that only contains the calculated COMs and reaction interfaces for visualization and comparison with the original PDB file. The input will be the returns of the previous function. Besides, the unit for the coordinates in PDB file is in Angstrom but not nm, so the value will be 10 times larger than that in NERDSS input files.

### OUTPUT - Create a 3D plot of each inputted chain

dtb\_PDB\_3D\_plot ()

Description: Generates a 3D plot indicaiting the spacial geometry of each simplified chain.

Parameters:

* Result

Acceptable value: Tuple (Must provide)

Description: The output result of function ‘real\_PDB\_separate\_read (FileName)’.

Sample: Result = result1

Example:

Clathrin.dtb\_PDB\_3D\_plot()

>>> *Shows graph of the interaction sites and COMs of each protein chain. I would show an example, but the function is broken lol.*

Long Description: This function will generate a 3D plot indicating the spacial geometry of each simplified chain. The solid lines of different colors are connecting the COM with interfaces within each chain; the black dotted line is connecting each pair of interfaces and the COMs are shown as solid points with their names above. To interact with the plot, other IDEs rather than Jupyter Notebook (such as VSCode) are recommended.

## Platonic Solid Self-assembly Input File Writing

Platonic solid self-assembly include 10 models, so that 10 separate functions are needed, which are shown in the following table:

|  |  |  |
| --- | --- | --- |
| Platonic Solid | Center-of-Mass Position | Name of Function |
| Tetrahedron (4-face) | Each Face | tetr\_face (radius, sigma) |
| Tetrahedron (4-face) | Each Vertex | tetr\_vert (radius, sigma) |
| Cube (6-face) | Each Face | cube\_face (radius, sigma) |
| Cube (6-face) | Each Vertex | cube\_vert (radius, sigma) |
| Octahedron (8-face) | Each Face | octa\_face (radius, sigma) |
| Octahedron (8-face) | Each Vertex | octa\_vert (radius, sigma) |
| Dodecahedron (12-face) | Each Face | dode\_face (radius, sigma) |
| Dodecahedron (12-face) | Each Vertex | dode\_vert (radius, sigma) |
| Icosahedron (20-face) | Each Face | icos\_face (radius, sigma) |
| Icosahedron (20-face) | Each Vertex | icos\_vert (radius, sigma) |

Description: Generates NERDSS input files (.inp and .mol files) for Platonic solid self-assembly system.

Parameters:

* radius (Float): It is the radius of the Platonic solid in **nm**, which is defined by the distance from the center of Platonic solid to each vertex.
* sigma (Float): It is the distance of each interface when a reaction takes place in **nm**.

Example:

tetr\_face(radius=10, sigma=1)

>>> *Creates parms.inp and .mol files for the self-assembly system for a tetrahedron with COM in the face*

# Analyzing NERDSS Outputs

Creates graphs, spreadsheets, pandas dataframes, 3D models, and more from the NERDSS outputs!

## Analyzing Histogram Files (General Functions)

**All histogram functions are methods of either the SingleHistogram object or MultiHistogram object. In this category, the function is included in both.**

### MAIN OBJECT – The Main Single-component Histogram Object (PLEASE READ!!)

SingleHistogram(FileName, FileNum, InitialTime, FinalTime, SpeciesName)

Description: An object that holds all of the data from a single species histogram file, and allows for easy interpretation of the data

Parameters:

* FileName (String)It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule listed below.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in seconds that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesName (String): It is the name of species that users want to examine, which should also be identical with the name written in the input (.inp and .mol) files.

Functions:

* Everything listed below in this section is a method of this object!

Example:

test\_histogram = ioNERDSS. SingleHistogram(FileName = "ioNERDSSPyPi\TestingFunctions\histogram\_single\_component.dat", FileNum = 1, InitialTime = 0.0, FinalTime = 1.00, SpeciesName = 'dode')

>>> *Initializes the test\_histogram SingleHistogram object for use in all future functions*

### MAIN OBJECT – The Main Multi-Component Histogram Object (PLEASE READ!!)

MultiHistogram(FileName, FileNum, InitialTime, FinalTime, SpeciesName)

Description: An object that holds all of the data from a multi species histogram file, and allows for easy interpretation of the data

Parameters:

* FileName (String)It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule listed below.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in **seconds** that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesList (List): The names of the species that are in the multi-histogram file, which should also be identical with the name written in the input (.inp and .mol) files.

Functions:

* Everything listed below in this section is a method of this object!

Example:

test\_histogram = ioNERDSS. MultiHistogram(FileName = "ioNERDSSPyPi\TestingFunctions\histogram\_multi\_component.dat", FileNum = 1, InitialTime = 0.0, FinalTime = 1.00, SpeciesList = [‘A’,’B’])

>>> *Initializes the test\_histogram SingleHistogram object for use in all future functions*

### DATAFRAME – stores count of each complex type for each timestep

hist\_to\_df (SaveCsv)

Description: Converts the raw .dat file to a data frame in python pandas package.

Parameters:

* SaveCsv (Bool, Optional = True): Whether the corresponding .csv file will also be saved as ‘histogram.csv’

Example:

test\_histogram multi\_hist\_to\_df(SaveCsv = False)

>>>

Time(s): A: 1. A: 1. B: 1. A: 1. B: 2. A: 1. B: 3. A: 1. B: 4. ... B: 2. B: 3. B: 4. B: 5. B: 6.

0 0.000 100 0 0 0 0 ... 0 0 0 0 0

1 0.001 86 4 1 0 0 ... 4 0 0 0 0

2 0.002 76 3 1 1 0 ... 5 0 0 0 0

Long Description: This function enables users to convert the raw .dat file to a data frame in python pandas package for multi-species system. Each column in the data frame includes the simulation time and selected occurrences of species during the simulation; each row is separated by a different simulation time.

### CSV – stores count of each complex type for each timestep

hist\_to\_csv ()

Description: This function enables users to convert the raw .dat file to a .csv file.

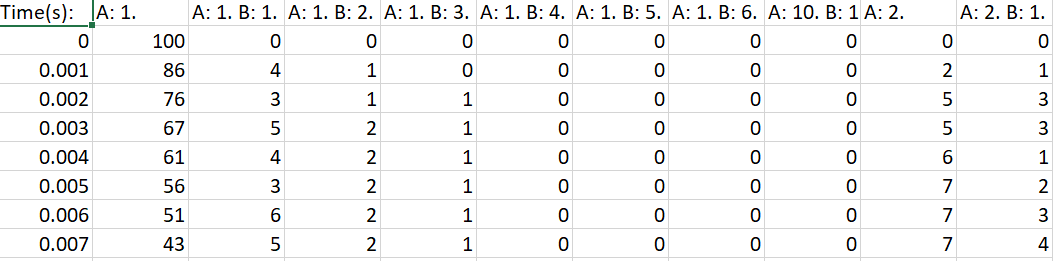
Parameters:

* None

Example:

test\_histogram. multi\_hist\_to\_csv()

>>>



Long Description: This function enables users to convert the raw .dat file to a .csv file for multi-species system. Each column in the data frame includes the simulation time and selected occurrences of species during the simulation; each row is separated by a different simulation time.

### LINE GRAPH – max count of protein species in a single complex at a time

line\_max\_complex\_size (SpeciesList, ShowFig, SaveFig)

Description: Creates a plot indicating maximum number of a specific protein species in single complex molecule during a certain time period.

Parameters:

* SpeciesName (Str, Optional-ish): Required if a multi-histogram file is input. It is the protein species that will be tracked.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

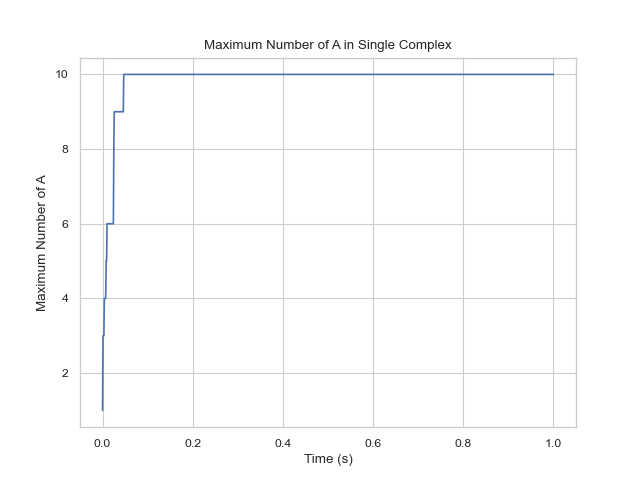
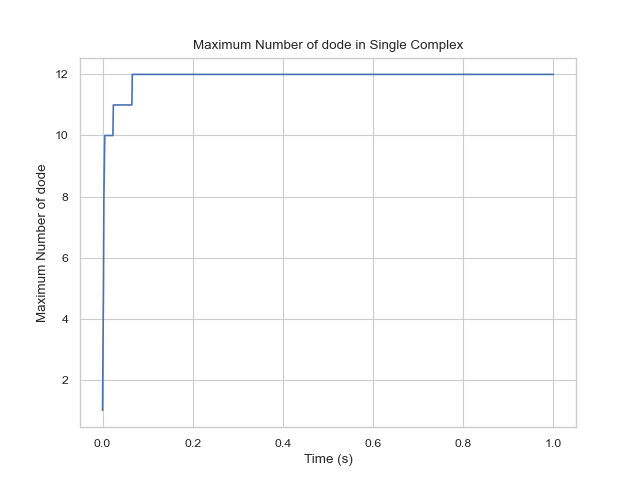
Example:

test\_histogran. line\_max\_complex\_size(ShowFig = True, SaveFig = False)

Or Multi-hist:

test\_histogran. line\_max\_complex\_size(SpeciesName=’A’,ShowFig = True, SaveFig = False)

>>>



Long Description: Will create a histogram where the X-axis is time, and Y-axis is the largest # of the tracked protein in a single complex

### LINE GRAPH – mean count of protein species in a single complex at a time

line\_mean\_complex\_size (SpeciesName, ShowFig, SaveFig)

Description: This function enables users to obtain a plot indicating mean number of a specific protein species in single complex molecule during a certain time period.

Parameters:

* SpeciesName (Str, Optional-ish): Required if a multi-histogram file is input. It is the protein species that will be tracked.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

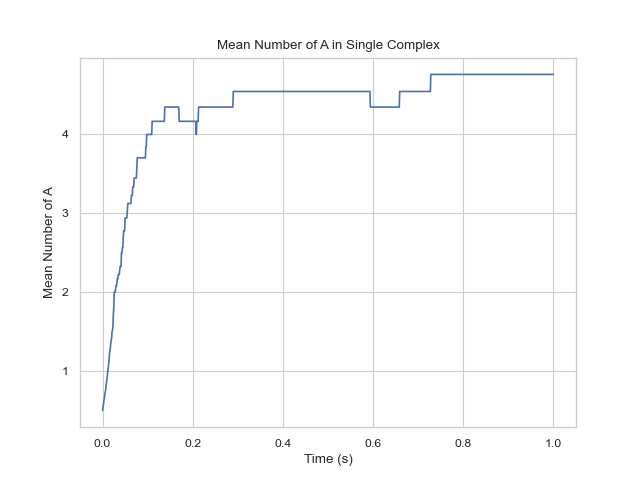
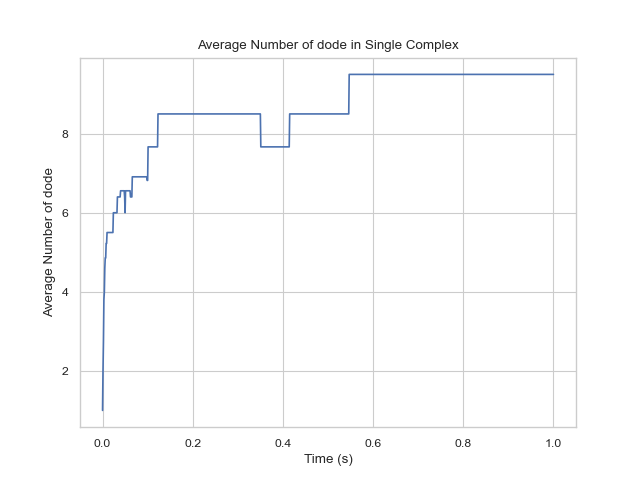
Example:

test\_histogram.line\_mean\_complex\_size(ShowFig = True, SaveFig = False)

Or multi-hist:

test\_histogram.line\_mean\_complex\_size(SpeciesName=’A’,ShowFig = True, SaveFig = False)

>>>



Long Description: Will create a histogram where the X-axis is time, and Y-axis is the average # of the tracked protein in a single complex

## Analyzing Single-component Histogram Files

**These are all methods of the SingleHistogram() object**

### HISTOGRAM – average number of each complex species size:

hist\_complex\_count (BarSize, ShowFig, SaveFig)

Description: Creates histogram of the average number of each type/size of complex species

Parameters:

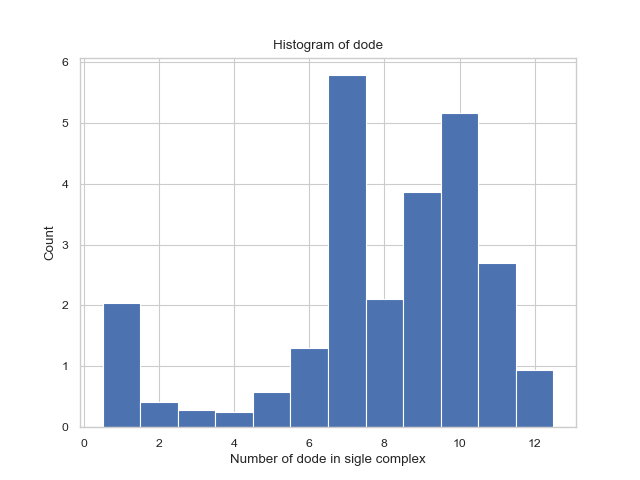
* BarSize (Int, Optional = 1): It is the size of each data bar in x-dimension. The x-axis will be separated evenly according to this number and the count of each size range will be summed up and shown together.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

Naming Rule: If single file is provided, the input file should be named as its original name (‘histogram\_complexes\_time.dat’); if multiple files are provided, the name of input file should also include serial number as ‘histogram\_complexes\_time\_X.dat’ where X = 1,2,3,4,5…

Example:

test\_histogram. hist\_complex\_count(BarSize = 1, ShowFig = True, SaveFig = False)

>>>



Long Description: Will create a histogram where the X-axis is each complex species type (each size), and Y-axis is the average count over the time period (Initial to Final)

### 3D HISTOGRAM – relative occurrence of each species over time:

hist\_3d\_complex\_count (TimeBins, xBarSize, ShowFig, SaveFig)

Description: Generates a 3D histogram representing the number of each type of complex over time.

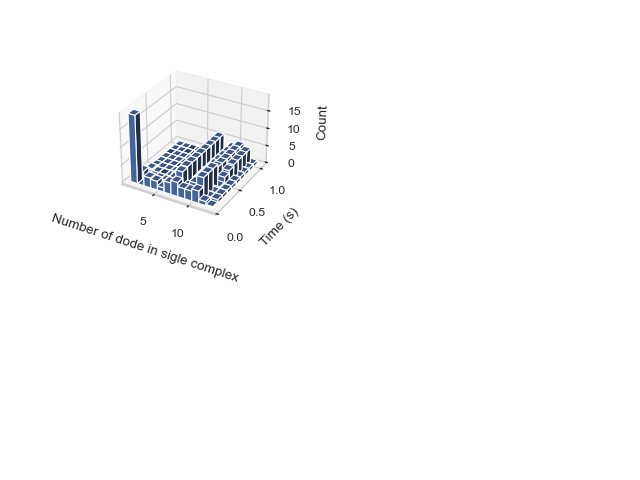
Parameters:

* TimeBins (Int): It is the number of bins that users want to divide the selected time period into. The value should be a positive integer.
* xBarSize (Int, Optional = 1): It is the size of each data bar in x-dimension. The x-axis will be separated evenly according to this number and the count of each size range will be summed up and shown together.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

Example:

test\_histogram.hist\_3d\_complex\_count(TimeBins=10, xBarSize = 1, ShowFig = True, SaveFig = False)

>>>



Long Description: This function enables users to generate a 3D histogram representing the number of monomers in a single complex as simulation time develops. The x-axis is the number of monomers, y-axis is the averaged time and z-axis is the relative occurrence probabilities.

### HEATMAP – average number of complexes at each time interval:

heatmap\_complex\_count (TimeBins, xBarSize, ShowFig, ShowMean, ShowStd, SaveFig)

Description: Generates a 2D histogram of numerical distribution of different complex sizes vs. time. corresponding time period is reached.

Parameters:

* TimeBins (Int): It is the number of bins that users want to divide the selected time period into. The value should be a positive integer.
* xBarSize (Int, Optional = 1): It is the size of each data bar in x-dimension. The x-axis will be separated evenly according to this number and the count of each size range will be summed up and shown together.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* ShowMean (Bool, Optional = False): Whether the corresponding mean value will be shown in the center of each box
* ShowStd (Bool, Optional = False): Whether the corresponding standard deviation value will be shown in the center of each box
* ShowFig (Bool, Optional = True): Whether the plot will be saved as a .png.

Example:

test\_histogram. heatmap\_species\_count(TimeBins = 10, xBarSize = 1, ShowFig = True, SaveFig = False, ShowMean=False, ShowStd=False)

>>>



Long Description: This function enables users to generate 2D histogram of numerical distribution of different N-mers vs. time. The x-axis is the distribution of number of monomers in single complex and y-axis is the time period. The color in each box indicates the number of corresponding N-mers when corresponding time period is reached.

### HEATMAP - total count of monomers in each complex size vs. time:

heatmap\_monomer\_count (TimeBins, xBarSize, ShowFig, ShowMean, ShowStd, SaveFig)

Description: Generates 2D histogram of total count of monomers in different complex species over time.

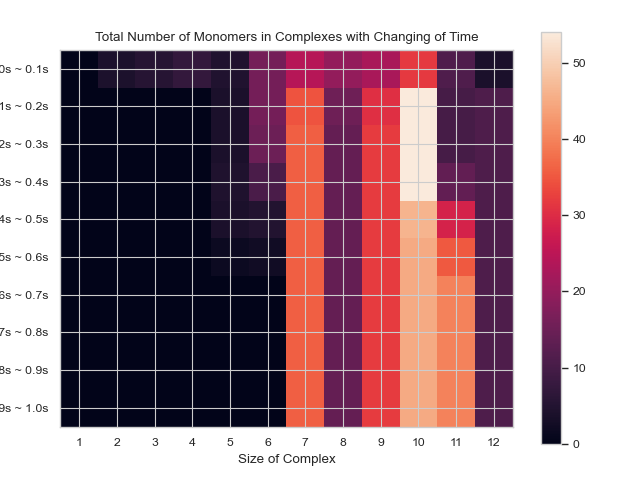
Parameters:

* TimeBins (Int): It is the number of bins that users want to divide the selected time period into. The value should be a positive integer.
* xBarSize (Int, Optional = 1): It is the size of each data bar in x-dimension. The x-axis will be separated evenly according to this number and the count of each size range will be summed up and shown together.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* ShowMean (Bool, Optional = False): Whether the corresponding mean value will be shown in the center of each box
* ShowStd (Bool, Optional = False): Whether the corresponding standard deviation value will be shown in the center of each box
* ShowFig (Bool, Optional = True): Whether the plot will be saved as a .png.

Example:

test\_function.heatmap\_monomer\_count(Timebins = 10, xBarSize = 1, ShowFig = True, SaveFig = False, ShowMean=False, ShowStd=False)

>>>



Long Description: This function enables users to generate 2D histogram of total count of monomers in different N-mers vs. time. The x-axis is the number of monomers in single complex and y-axis is the time period. The color in each box indicates the total number of corresponding monomers in N-mers when corresponding time period is reached.

### HEATMAP - fractions of original monomers in each complex species vs. time:

heatmap\_monomer\_fraction (TimeBins, xBarSize, ShowFig, ShowMean, ShowStd, SaveFig)

Description: Generates 2D histogram of fraction of original monomers in different complex species over time.

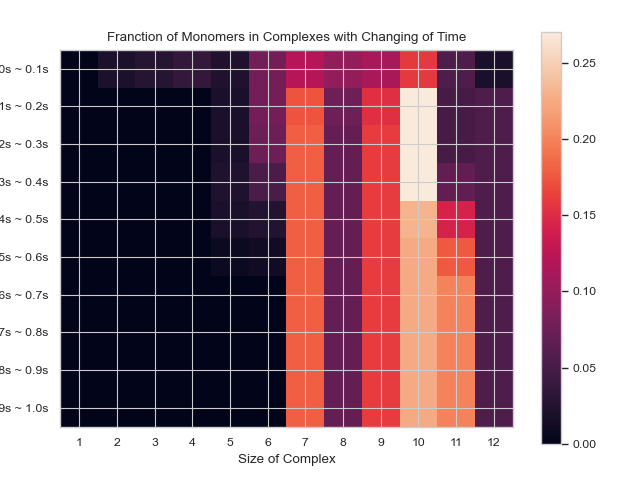
Parameters:

* TimeBins (Int): It is the number of bins that users want to divide the selected time period into. The value should be a positive integer.
* xBarSize (Int, Optional = 1): It is the size of each data bar in x-dimension. The x-axis will be separated evenly according to this number and the count of each size range will be summed up and shown together.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* ShowMean (Bool, Optional = False): Whether the corresponding mean value will be shown in the center of each box
* ShowStd (Bool, Optional = False): Whether the corresponding standard deviation value will be shown in the center of each box
* ShowFig (Bool, Optional = True): Whether the plot will be saved as a .png.

Example:

test\_function.heatmap\_monomer\_fraction(TimeBins = 10, xBarSize = 1, ShowFig = True, SaveFig = False, ShowMean=False,ShowStd=False)

>>>



Long Description: This function enables users to generate 2D histogram of fractions of monomers forming different N-mers vs. time. The x-axis is the number of monomers in single complex and y-axis is the time period. The color in each box indicates the fraction of monomers forming corresponding N-mers when corresponding time period is reached.

## Analyzing Multi-component Histogram Files

**These are all methods of the MultiHistogram() object**

### HISTOGRAM – Frequency of each complex size

hist\_complex\_count (FileName, FileNum, InitialTime, FinalTime, SpeciesList, BinNums, ExcludeSize, ShowFig, SaveFig)

Description: Creates a general histogram of total size of complex or selected species inside each complex for a multi-species system.

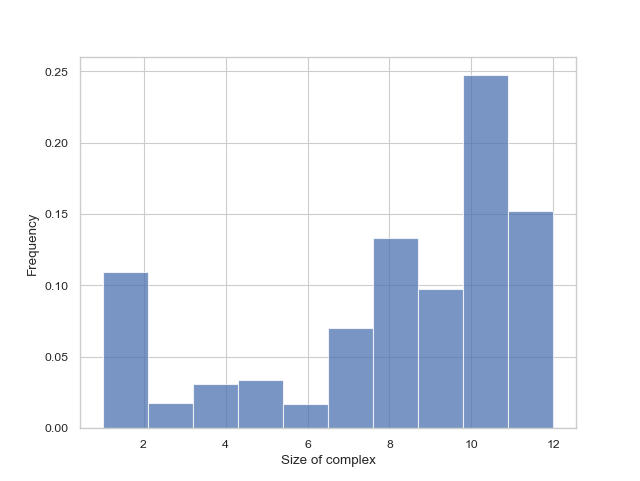
Parameters:

* BinNums: (Int, Optional = 10): Number of bins in the histogram.
* ExcludeSize (Int, Optional = 0): In the generated plot, the number of monomers in the complex that are no larger than this number will be excluded and will not be considered into the average calculation.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

Example:

test\_histogram.hist\_complex\_count(BinNums = 10, ShowFig = True, SaveFig = False)

>>>



Long Description: This function enables users to plot general histogram of total size of complex or selected species inside each complex for a multi-species system. It will also analyze multiple input files and show the result along with error bar. The x-axis is the size of selected species or total number of monomers, and the y-axis is the average number of counts for corresponding size.

### STACKED HISTOGRAM – Counts of complex species with certain protein compositions

stack\_hist\_complex\_count(xAxis, DivideSpecies, DivideSize, BarSize, ExcludeSize, ShowFig, SaveFig)

Description: Plots general histogram of total size of selected species for a multi-species system. Each bar is split into three stacked bars which represent the size distribution of another selected species compared to a desired input.

Parameters:

* xAxis (String): It indicates the species shown on the x-axis. If xAxis is included inside SpeciesList, the x-axis will only show the number of selected components. The other input is ‘tot’, which represents the x\_axis will count all species in a single complex.
* DivideSpecies (String): It indicates the name of the species that users want to separate by size.
* DivideSize (Int): This is the value that separates the size of the dissociate complex, for example, if DivideSize = 5, that means the dissociate events are classified as ‘DivideSpecies size < 5’, ‘DivideSpecies size = 5’ and ‘DivideSpecies size > 5’.
* BarSize (Int, Optional = 1): It is size of each data bar in x-dimension. The x-axis will be separated evenly according to this number and the count of each size range will be sum up and shown together.
* ExcludeSize (Int, Optional = 0): In the generated plot, the number of monomers in the complex that are no larger than this number will be excluded and will not be considered into the average calculation.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

Example:

test\_histogram. multi\_hist\_stacked(xAxis = "A", DivideSpecies = "B", DivideSize = 5, BarSize = 1, ShowFig = True, SaveFig = False)

>>>



Long Description: This function enables users to plot general histogram of total size of complex or selected species for a multi-species system. Each bar is split by three stacked bars which represent the size distribution of another selected species compared to a desired input. It will also analyze multiple input files and show the result along with error bar. The x-axis is the size of selected species or total number of monomers, and the y-axis is the average number of counts for corresponding size.

### HEATMAP – Average count of each complex composition over simulation time

heatmap\_ complex\_dist(xAxis, yAxis, xBarSize, yBarSize, ShowFig, ShowMean, ShowStd, SaveFig)

Description: Creates a heatmap during a certain time period representing the distribution of size of selected species.

Parameters:

* xAxis (String): It indicates the species shown on the x-axis. If xAxis is included inside SpeciesList, the x-axis will only show the number of selected components.
* yAxis (String): It indicates the species shown on the x-axis. If yAxis is included inside SpeciesList, the x-axis will only show the number of selected components.
* xBarSize (Int, Optional = 1): It is size of each data bar in x-dimension. The x-axis will be separated evenly according to this number and the count of each size range will be sum up and shown together.
* yBarSize (Int, Optional = 1): It is size of each data bar in y-dimension. The y-axis will be separated evenly according to this number and the count of each size range will be sum up and shown together.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* ShowMean (Bool, Optional = False): Whether the corresponding mean value will be shown in the center of each box
* ShowStd (Bool, Optional = False): Whether the corresponding standard deviation value will be shown in the center of each box
* ShowFig (Bool, Optional = True): Whether the plot will be saved as a .png.

Example:

test\_histogram.heatmap\_complex\_dist(xAxis=’A’, yAxis=’B’, xBarSize=1, yBarSize=1, ShowFig = True, ShowMean=False, ShowStd=False, SaveFig = False)

>>>



Long Description: This function enables users to generate a heatmap during a certain time period representing the distribution of size of selected species. The x and y axis are both desired individual components and the color of each square represents the relative occurrence probability of complex of corresponding size.

### 3D HISTOGRAM - Average count of each complex composition over simulation time

hist\_3D\_complex\_dist(xAxis, yAxis, xBarSize, yBarSize, ShowFig, SaveFig)

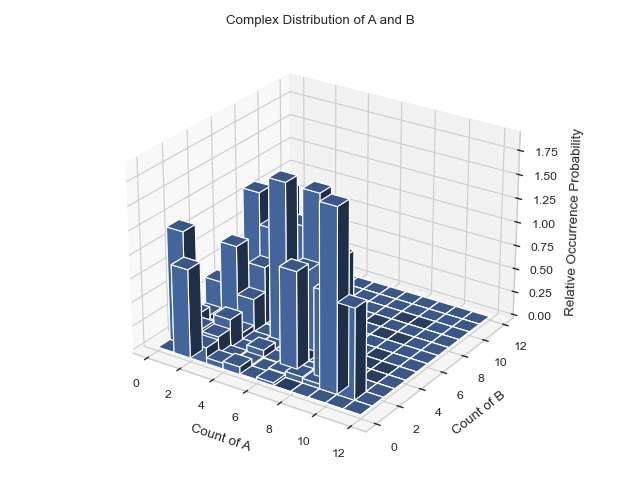
Description: Generates a 3D histogram during a certain time period representing the distribution of size of selected species.

Parameters:

* xAxis (String): It indicates the species shown on the x-axis. If xAxis is included inside SpeciesList, the x-axis will only show the number of selected components.
* yAxis (String): It indicates the species shown on the x-axis. If yAxis is included inside SpeciesList, the x-axis will only show the number of selected components.
* xBarSize (Int, Optional = 1): It is size of each data bar in x-dimension. The x-axis will be separated evenly according to this number and the count of each size range will be sum up and shown together.
* yBarSize (Int, Optional = 1): It is size of each data bar in y-dimension. The y-axis will be separated evenly according to this number and the count of each size range will be sum up and shown together.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* ShowFig (Bool, Optional = True): Whether the plot will be saved as a .png.

Example:

test\_histogram.hist\_3D\_complex\_dist(xAxis=’A’, yAxis=’B’, xBarSize=1, yBarSize=1, ShowFig = True, SaveFig = False)



Long Description: Generates a 3D histogram during a certain time period representing the distribution of size of selected species. The x and y axis are both desired individual components and the height of each column represents the relative occurrence probability of complex of corresponding size.

## Analyzing Transition Matrix Files

### \*Incomplete LINE PLOT - free energy among different size of complexes:

free\_energy (FileName, FileNum, InitialTime, FinalTime, SpeciesName, ShowFig, SaveFig)

Description: The plot indicates the change in free energy in selected time period among different size of complexes. The x-axis is the size of complex and the y-axis is the free energy calculated as in the unit of , where the refers to the probability of occurrence of the number of times N-mer is counted (including association and dissociation). If multiple input files are given, the output plot will be the average value of all files and an error bar will also be included.

Parameters:

* FileName (String): It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in seconds that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesName (String): It is the name of species that users want to examine, which should also be identical with the name written in the input (.inp and .mol) files.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

### \* Incomplete LINE PLOT - symmetric associate probability:

associate\_prob\_symmetric(FileName, FileNum, InitialTime, FinalTime, SpeciesName, DivideSize, ShowFig, SaveFig)

Description: This line plot represents the probability of association between complexes of different sizes and other complexes of different sizes. The x-axis is the size of the complex and y-axis is the associate probability. Three lines will exist in the line graph, representing associating to complexes of sizes less than, equal to, or greater than the specified size, respectively. 'Symmetric' in the function name means that for the associate reaction, both sizes of complexes are counted as associating events symmetrically, for example, if an associate event occurs where a trimer associates to a tetramer as a heptamer, then this event is counted twice, which are trimer associates to tetramer and tetramer associates to trimer. If multiple input files are given, the output plot will be the average value of all files and an error bar will also be included.

Parameters:

* FileName (String): It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in seconds that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesName (String): It is the name of species that users want to examine, which should also be identical with the name written in the input (.inp and .mol) files.
* DivideSize (int, Optional = 2): This is the value that distinguishes the size of the associate complex, for example, if DivideSize = 2, that means the associate events are classified as ‘associate size < 2’, ‘associate size = 2’ and ‘associate size > 2’.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

### \* Incomplete LINE PLOT - asymmetric associate probability:

associate\_prob\_asymmetric(FileName, FileNum, InitialTime, FinalTime, SpeciesName, DivideSize, ShowFig, SaveFig)

Description: This line plot represents the probability of association between complexes of different sizes and other complexes of different sizes. The x-axis is the size of the complex and y-axis is the associate probability. Three lines will exist in the line graph, representing associating to complexes of sizes less than, equal to, or greater than the specified size, respectively. 'Asymmetric' in the function name means that for the associate reaction, only the complexes of smaller size associating to the larger one is counted as associate event asymmetrically, for example, if an associating event occurs where a trimer associates to a tetramer as a heptamer, then this event is counted only once, which is a trimer associates to tetramer. If multiple input files are given, the output plot will be the average value of all files and an error bar will also be included.

Parameters:

* FileName (String): It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in seconds that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesName (String): It is the name of species that users want to examine, which should also be identical with the name written in the input (.inp and .mol) files.
* DivideSize (int, Optional = 2): This is the value that distinguishes the size of the associate complex, for example, if DivideSize = 2, that means the associate events are classified as ‘associate size < 2’, ‘associate size = 2’ and ‘associate size > 2’.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

### \* Incomplete LINE PLOT - symmetric dissociate probability:

dissociate\_prob\_symmetric (FileName, FileNum, InitialTime, FinalTime, SpeciesName, DivideSize, ShowFig, SaveFig)

Description: This line plot represents the probability of dissociation of complexes of different sizes into other complexes of different sizes. The x-axis is the size of the complex and y-axis is the dissociate probability. Three lines will exist in the line graph, representing dissociating to complexes of sizes less than, equal to, or greater than the specified size, respectively. 'Symmetric' in the function name means that for the dissociate reaction, both sizes of complexes are counted as dissociating events symmetrically, for example, if an dissociate event occurs where a heptamer dissociates into a tetramer and a trimer, then this event is counted twice, which are heptamer dissociates to tetramer and heptamer dissociates to trimer. If multiple input files are given, the output plot will be the average value of all files and an error bar will also be included.

Parameters:

* FileName (String): It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in seconds that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesName (String): It is the name of species that users want to examine, which should also be identical with the name written in the input (.inp and .mol) files.
* DivideSize (int, Optional = 2): This is the value that distinguishes the size of the associate complex, for example, if DivideSize = 2, that means the associate events are classified as ‘associate size < 2’, ‘associate size = 2’ and ‘associate size > 2’.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

### \* Incomplete LINE PLOT - asymmetric dissociate probability:

dissociate\_prob\_asymmetric (FileName, FileNum, InitialTime, FinalTime, SpeciesName, DivideSize, ShowFig, SaveFig)

Description: This line plot represents the probability of dissociation of complexes of different sizes into other complexes of different sizes. The x-axis is the size of the complex and y-axis is the dissociate probability. Three lines will exist in the line graph, representing dissociating to complexes of sizes less than, equal to, or greater than the specified size, respectively. 'Asymmetric' in the function name means that for the dissociate reaction, only the complexes of smaller size dissociating from the original one is counted as dissociate event asymmetrically, for example, if an dissociate event occurs where a heptamer dissociates into a tetramer and a trimer, then this event is counted only once, which is heptamer dissociates to trimer. If multiple input files are given, the output plot will be the average value of all files and an error bar will also be included.

Parameters:

* FileName (String): It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in seconds that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesName (String): It is the name of species that users want to examine, which should also be identical with the name written in the input (.inp and .mol) files.
* DivideSize (int, Optional = 2): This is the value that distinguishes the size of the associate complex, for example, if DivideSize = 2, that means the associate events are classified as ‘associate size < 2’, ‘associate size = 2’ and ‘associate size > 2’.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

### \* Incomplete LINE PLOT – Growth probability for each complex size:

growth\_prob (FileName, FileNum, InitialTime, FinalTime, SpeciesName, ShowFig, SaveFig)

Description: This line plot indicates the probability of growth in size for different sizes of complexes. The x-axis is the size of complexes, and the y-axis is the growth probability. If multiple input files are given, the output plot will be the average value of all files and an error bar will also be included.

Parameters:

* FileName (String): It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in seconds that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesName (String): It is the name of species that users want to examine, which should also be identical with the name written in the input (.inp and .mol) files.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

### LINE PLOT – Average lifetime for each complex type:

complex\_lifetime(FileName, FileNum, InitialTime, FinalTime, SpeciesName, ShowFig, SaveFig)

Description: This line plot indicates the average lifetime for each complex size.

Parameters:

* FileName (String): It is the path to the ‘.dat’ file, which is usually named as ‘histogram\_complexes\_time.dat’, representing the histogram data to be analyzed.
* FileNum (Int): It is the number of the total input file. If multiple files are provided, their names should obey the naming rule.
* InitialTime (Float): It is the initial time in **seconds** that will be examined. Must be smaller than FinalTime and bigger than start time in the file.
* FinalTime (Float): It is the final time in seconds that will be examined. Must be bigger than InitialTime and smaller than max time in the file.
* SpeciesName (String): It is the name of species that users want to examine, which should also be identical with the name written in the input (.inp and .mol) files.
* ShowFig (Bool, Optional = True): Whether the plot will be shown.
* SaveFig (Bool, Optional = False): Whether the plot will be saved as a ‘.png’ file

Example:

IoNERDSS. complex\_lifetime(FileName = "ioNERDSSPyPi\TestingFunctions\\transition\_matrix\_time.dat", FileNum = 1, InitialTime = 0.0, FinalTime = 1.00, SpeciesName = 'dode', ShowFig = True, SaveFig = False)

>>>



Long Description: This line plot indicates the lifetime for different sizes of complexes. The x-axis is the size of complexes, and the y-axis is its average lifetime in unit of second. If multiple input files are given, the output plot will be the average value of all files and an error bar will also be included.

## Locating position for certain size of complexes by PDB/restart file

### By PDB file:

filter\_PDB\_op\_no\_restart (FileNamePdb, NumDict, FileNameInp, BufferRatio)

Description: This function enables users to locate specific complexes of certain size from a PDB file after simulation. The result will be output as a new file named “output\_file.pdb” containing only the desired complex.

Parameters:

* FileNamePdb (str): The path to the PDB file, which is usually the last frame of the simulation.
* NumDict (dictionary): A dictionary that holds the requested number of protein types in a complex
* FileNameInp (str): The path to the '.inp' file, which usually stores the reaction information.
* BufferRatio (float, optional = 0.01): The buffer ratio used to determine whether two reaction interfaces can be considered as bonded.

Example:

IoNERDSS. filter\_PDB\_op\_no\_restart(FileNamePdb = "ioNERDSSPyPi\TestingFunctions\\nerdss\_output.pdb", NumDict={"dod":9}, FileNameInp="ioNERDSSPyPi\TestingFunctions\parm.inp")

>>> *Output\_file.pdb that includes only proteins in complexes of the selected size*

...

ATOM 19 COM dod 3 301.720 116.470 306.361 0 0CL

ATOM 20 lg1 dod 3 315.636 126.000 315.231 0 0CL

ATOM 21 lg2 dod 3 312.386 125.024 293.086 0 0CL

....

### By ‘restart.dat’ file:

filter\_PDB\_op\_restart(FileNamePdb, NumDict, FileNameRestart)

Description: This function enables users to locate specific complexes of certain size from a PDB file along with ‘restart.dat’ file after simulation. The result will be output as a separated file named “output\_file.pdb” containing only the desired complex.

Parameters:

* FileNamePdb (str): The path to the PDB file, which is usually the last frame of simulation.
* NumDict (dictionary): A dictionary that holds the requested number of protein types in a complex
* FileNameRestart (str): The path to the 'restart.dat' file. Defaults to 'restart.dat'.

Example:

IoNERDSS. filter\_PDB\_op\_restart(FileNamePdb = "ioNERDSSPyPi\TestingFunctions\\nerdss\_output.pdb", NumDict={"dod":9}, FileNameRestart="ioNERDSSPyPi\TestingFunctions\restart.dat")

>>> *Output\_file.pdb that includes only proteins in complexes of the selected size*

...

ATOM 19 COM dod 3 301.720 116.470 306.361 0 0CL

ATOM 20 lg1 dod 3 315.636 126.000 315.231 0 0CL

ATOM 21 lg2 dod 3 312.386 125.024 293.086 0 0CL

ATOM 22 lg3 dod 3 294.395 112.226 289.287 0 0CL

...

Additional Info: The advantage of reading the 'restart.dat' file is that the file directly stores the binding information of each complex in the system and can be used directly, so the function runs faster; however, the function is not universal, if the 'restart.dat ' file's write logic changes, then this function will no longer work.

## Analyzing .xyz files

### CSV – creates spreadsheet of protein locations

xyz\_to\_csv (FileName, LitNum):

Description: This function enables users to convert the output .xyz file by NERDSS simulation into a .csv file of a specific or entire time frame.

Parameters:

* FileName (String): It is the path to the .xyz file, which is usually names as ‘trajectory.xyz’.
* LitNum (Int, Optional = -1)

Description: It is the number of literation user desire to examine. If the input is -1, the function will extract the entire literation.

Example:

IoNERDSS. xyz\_to\_csv(FileName="ioNERDSSPyPi\TestingFunctions\\trajectory.xyz", LitNum=-1)

>>>



Long Description: This function enables users to convert the output .xyz file by NERDSS simulation into a .csv file of a specific or entire time frame. The generated csv file will contain 5 columns, including number of literation, species name, x, y, and z coordinates.

### DATAFRAME – creates dataframe of protein locations

xyz\_to\_df (FileName, LitNum, SaveCsv):

Description: This function enables users to convert the output .xyz file by NERDSS simulation into a pandas.DataFrame of a specific or entire time frame. The generated csv file will contain 5 columns, including number of literation, species name, x, y and z coordinates.

Parameters:

* FileName (String): It is the path to the .xyz file, which is usually names as ‘trajectory.xyz’.
* LitNum (Int, Optional = -1)

Description: It is the number of literation user desire to examine. If the input is -1, the function will extract the entire literation.

* SaveCsv (Bool, Optional = True): Whether the corresponding .csv file will be saved

Example:

IoNERDSS. xyz\_to\_df(FileName="ioNERDSSPyPi\TestingFunctions\\trajectory.xyz", LitNum=-1, SaveCsv = False)

>>> literation name x y z

0 0 ap 87.420620 -270.109172 -203.661987

1 0 ap 88.081526 -271.052470 -205.297038

2 0 ap 86.759715 -269.165874 -202.026936

3 0 ap -58.647113 277.528515 -353.236112

...

### MATRIX - tracks the trajectory of specific protein(s)

traj\_track (FileName, SiteNum, MolIndex)

Description: racks the COM coordinate changing of one or more molecule.

Parameters:

* FileName (String): It is the path to the .xyz file, which is usually names as ‘trajectory.xyz’.
* SiteNum (Int): This is the total number of COM and interfaces of a single molecule. For example, if a molecule possesses 1 COM and 5 interfaces, the SiteNum value should be 6.
* MolIndex (List with Int elements): This is the index of molecule users desired to track. The number in the list should be no smaller than 1.

Example:

IoNERDSS. traj\_track(FileName="ioNERDSSPyPi\TestingFunctions\\trajectory.xyz", SiteNum=3, MolIndex = [1,4,10])

>>>

[[[87.42062, -270.109172, -203.661987], [40.873538, 168.96348, -497.993163]],

[[74.407358, 51.461467, -242.958456], [187.824563, 325.913499, -497.993163]],

[[20.608487, 330.919045, -182.061499], [-27.367719, 330.945162, -497.993163]]]

Long Description: This function enables users to track the COM coordinate changing of one or more molecule. The return will be a 2D matrix with the size of the number of literation times the number of desired molecules.