Component Specification

Software components

- User interface: provides a clear means for the user to input the required data and desired outputs. The user will provide names and values as inputs, specifying protomers, concentrations, affinities, labels, and kinetics rate constants if desired. This information will be input as a python dictionary, with examples in the notebook to demonstrate the required formatting and guide the user. The user will then be able to request a type of simulation from the system in the form of a method call from the class, and the system will simulate and output the data. The user interface will consist of clear instructions within the notebook, examples to work off of, as well as prompts before and after each interaction that the user makes to steer the user to next steps.
- Protomer network object: this class will contain all of the necessary information about the protomer network that the user inputs as a dictionary. It will also contain all possible simulations that the user can run for the network, as methods. These methods will run simulations and output the information that the user requests.
- Simulate equilibrium species distribution: this method will use the protomer network from the class to run a simulation and return the distributions of all species at equilibrium. It will take the subset of protomers to use in the simulation as an input list, and use their concentrations and affinities that are already stored in a dictionary to simulate the equilibrium species distribution. It will output this distribution in the form of each species and its concentration, as its own dictionary. If kinetics rate constants are not provided, the simulation will assume diffusion-limited k_{on} rates, as the required output from this method is only the system at equilibrium. If the user provides kinetics rate constants for the protomer species interactions, these constants will be included in the simulation, to provide more accurate data of the equilibration of the system. These graphs of species distributions over time (or just the equilibrium species distributions) can then be plotted using the visualization module.
- Visualization module: this method of the class will allow the user to visualize simulation outputs using seaborn, using data stored for each simulation.

Interactions to accomplish use cases

For the first use case where the user wants to predict the equilibrium distribution of species for a given network of protomers, the user will interact with the system through the user interface. The interface will prompt the user to input all protomers, concentrations, and affinities to initialize the network through the protomer network object. The user will do this by inputting a python dictionary with the required information. Once this is done, the interface will prompt the user to select the desired simulation, which in this case is the equilibrium distribution of species. The user will call the method as an input. When the method is called, it will use the stored python dictionary as an input to run the simulation, and output the basic results to the user in the form of another python dictionary with the equilibrium species distribution. The full data of the simulation will be stored in the protomer network object. If visualizations are specified, the visualization module will use the stored simulation data as an input to plot the equilibrium distributions and/or the time series data of the simulation, and output these visualizations to the user in the notebook as seaborn plots with titles and labels. The

visualization module will be called by the user as a separate method. If specified, both the output data and visualization plots can be saved to a desired file path and file format to be used outside of the tool.

Preliminary plan

1. Build protomer network object, which consists of a class containing all data structures that will be required to run simulations
2. Implement the most basic ‘simulate equilibrium species distribution’ method, which will be called by other more elaborate methods
3. Build unit tests for this first method to ensure it is working properly and returning correct outputs
4. Script a user interface on top of the protomer network object that will allow for user interactions with the class and a simple user experience
5. Build out additional methods to allow for more complex simulations and outputs. For each additional method:
    a. Write and run test cases specific to the method
    b. Expand the user interface to include the new functionality
    c. Build system tests to test the interactions of components and make sure the tool as a whole runs as intended