Functional Specification

Background

In the Baker lab, we have developed many *de novo* protein heterodimers. Each protomer (half) of the heterodimer will bind with some affinity to its cognate binding partner, but may also be cross-reactive and bind to other protomer species. When multiple protomer species are combined, protomers can competitively bind and displace one another, as binding is monovalent and occurs at a shared interface. The promiscuity of these protomers enables a form of computation using networks of interacting protomers, where the addition of a new protomer results in a redistribution of monomer and dimer species. When coupled to a split reporter reconstituted by the association of certain dimer species, the behaviour of the network can be read out as a signal. These networks can be used to create logic gates, where protomer additions are inputs and the signal is interpreted as a binary output. These logic gates can be designed on paper by looking at the affinities of different interacting pairs and reasoning through the competition that will occur and the resulting signal, but this method is not scalable and quickly becomes complicated. Therefore, a computational method to model the networks provided the affinities of the protomers and simulate the equilibrium distribution of species and signal output would be useful in applying the technology.

User profile

This tool will be designed for a research scientist user. The user will likely have some background in coding, but should not need to be proficient to use the tool. The user should be able to run a notebook and call functions within a class with instructions provided.

Use cases

1. A first use case is the one-off case, where the user simply wants to predict the equilibrium distribution of species for a given network of protomers. The user provides the identities and concentrations of all protomers and the affinity of each interaction as prompted in a python dictionary, and the system simulates the equilibrium state (as a function call within a class) to return the distribution across all species present.
2. A second use case is for logic gate construction, where the user wants to predict the binary logic output of a network consisting of a pool of protomers and a set of input protomers. This could be to predict the logic behavior of a single network setup, or to couple to another script to more systematically explore logic gate creation for some set of protomers. In either scenario, the user provides the identities and concentrations of all protomers, labels assigning protomer inputs (and input identity), and labels designating which dimer species correspond to the reporter signal. The system simulates the equilibrium distribution of species for all combinations of inputs, and returns a vector containing either the raw or binary signal for each input combination, which summarizes the logic gate response of the protomer network. In this use case, the user is more familiar with the tool, and so the printouts of the tool are suppressed so that the tool can be used at scale for its methods.

GitHub repository

https://github.com/idlutz/protomer-network-distributions