Modeling and Characterizing BMP-2 Protein Binders for Fracture Regeneration

by

KARLY FEAR

A THESIS

Presented to the Department of Biology
and the Robert D. Clark Honors College
in partial fulfillment of the requirements for the degree of
Bachelor of Science

May 2022

An Abstract of the Thesis of

Karly Fear for the degree of Bachelor of Science
in the Department of Biology to be taken May 2022

Title: Modeling and Characterizing BMP-2 Protein Binders for Fracture Regeneration

Approved: _____*Parisa Hosseinzadeh, PhD*_______
Primary Thesis Advisor

Critically-sized bone defects experience delayed regeneration correlated to deficient protein signaling at the injury site. Delivering the potent growth factor bone morphogenetic protein-2 (BMP-2) to the defect is a promising clinical intervention to augment fracture healing in such cases. The release kinetics of BMP-2 from a hydrogel scaffold play a crucial role in the success of therapeutic delivery in terms of improved healing outcome. For instance, rapid BMP-2 release causes suboptimal bone formation. Affinity interactions between BMP-2 and binder proteins offer a mechanism to fine-tune the spatial placement and timing of exogenous BMP-2 bioactivity over the course of bone healing.

Experimental selection techniques may identify such binder proteins from large protein libraries. However, experimental pipelines fail to reveal structural details of the protein-protein interaction and explore only a small subset of the possible sequence landscape for binding proteins. The computational design route is therefore an attractive strategy to rationally engineer a BMP-2 delivery control mechanism. In order to slow BMP-2 release, I first sought to directionally modify experimentally selected binding proteins with computational modeling and mutagenesis. Next, I generated de novo protein binders hypothesized to retain BMP-2 via affinity interactions when conjugated to the hydrogel scaffold. I approached this challenge with well-established computational protein design methodologies to yield binding proteins with target surface site-specificity. I then developed a machine learning model to predict binding affinity for computationally modeled proteins. Finally, I sought to experimentally characterize my designed binders using yeast surface display and flow cytometry to validate in vitro that my computational pipeline generates functional designs, appropriate for controlled BMP-2 delivery.

Acknowledgements

I would like to thank Professors Parisa Hosseinzadeh, Marian Hettiaratchi, Keat Ghee Ong, and Robert Guldberg for helping me to fully examine this interdisciplinary project from the various perspectives of computational protein engineering and regenerative therapy.

I am so grateful for the unwavering guidance and encouragement of my family, friends, and especially my PI and primary thesis advisor, Parisa, as well as the patience and persistence of our collaborators Jonathan Dorogin and my second thesis committee member, Marian Hettiaratchi. Thank you to all the generous and stimulating members of the Hettiaratchi, Ong, Guldberg, and Plesa labs who inspired and aided in this project. The reported work was supported by the Knight Campus for Accelerating Scientific Impact and the University of Oregon Vice President of Research and Innovation.

I would also like to thank my CHC reader Trond Jacobsen for his critical reading and feedback in this thesis project as well as the faculty and advisors of the Clark Honors College and the offices of CURE and UROP who have also supported my academic and research journey.

Table of Contents

Introduction

Non-union fractures require medical intervention

Each year, over 630,000 people suffer from non-union bone fractures in the US. Accounting for about 10% of fractures, non-union or delayed fractures result in the incomplete bridging of bone due to the extent and size of the bone and soft tissue damage [1]. These critically-sized injuries require further medical intervention due to the impairment of protein signaling pathways for regeneration. Researchers are increasingly investigating the delivery of therapeutic proteins or biologics to promote local regeneration via protein expression and signaling at the injury site [2]. Biologics may be native proteins that have endogenous bioactivity.

The family of growth factors known as bone morphogenetic proteins (BMPs) offer potential as regenerative biologics. Members of the endogenous human BMP family, such as BMP-2, have various functions in development and tissue regeneration. To be effectively therapeutic in treating non-union bone fractures, exogenous BMP-2 needs to be delivered to the site of injury at a high dose over the weeks and months of bone proliferation: the stage of healing that follows initial inflammation and precedes bone remodeling. BMP-2 acts during the proliferative stage of healing by signaling progenitor cells to differentiate into osteoblasts, or bone mineralizing cells [3].

In order to accomplish effective BMP-2 biologic therapy, the protein must be delivered with optimal spatiotemporal control. Collagen sponges used for delivering BMP-2 locally to fractures are limited by their low retention rate. Rather than a slow and steady release over several weeks, the sponge burst releases large doses of BMP-2 in the first 24 hours of delivery. The rapid release of biologic when employing a collagen sponge delivery vehicle necessitates the use of above physiological doses of growth factor, leading to heterotopic ossification, a condition of abnormal superficial bone growth. To prevent this condition, BMP-2 needs to be delivered in a more gradual manner, a method called controlled release. To achieve slower, controlled release, researchers developed a novel approach utilizing designed hydrogels. Hydrogel biomaterials like hyaluronic acid and alginate have been explored for their cross-linking properties, which may control the release dynamics of the biologic. Alternatively, hydrogels may be fabricated with BMP-2 binding proteins, thereby mediating biologic delivery with affinity interactions [4]. In this method, a small protein is covalently linked to the hydrogel and allowed to interact with soluble BMP-2 during fabrication. By using small proteins that are known to bind to BMP-2 with a particular binding affinity, the growth factor is retained within the hydrogel scaffold more reliably, preventing burst release and sequestering endogenous growth factor for a naturally amplified therapeutic effect.

The success of this approach relies heavily on obtaining BMP-2 binders with certain binding characteristics: proteins that bind BMP-2 moderately–in the low μM k_d range–may afford a slow release rate or k_{off}. These small protein binders can be obtained using magnetic bead sorting from a highly diverse affibody library. An affibody is a 58 amino acid long three helical protein scaffold with 13 variable positions (fig. 1). After magnetic bead sorting, affibodies that interact with the target protein undergo high- throughput analysis using yeast surface display and flow cytometry, or fluorescence activated cell sorting (FACS). The selected affibodies require further characterization and optimization to reach the desired binding affinity or k_d [5]. This experimental pipeline can achieve small protein binders with nM affinity, but fails to reveal any structural information about the site or orientation of binding. Moderate and low affinity binders identified by magnetic activated cell sorting (MACS) are susceptible to low specificity, which may lead to adverse off-target effects. Downstream optimization of protein binders without the aid of structural information is an inefficient and intensive experimental process, when compared to pipelines that utilize computational protein modeling.

Fortunately, recent advances in the fields of computation and protein biophysics have eclipsed to enable researchers to obtain accurate structural models of biomolecules. Such molecular models can be used to investigate the intermolecular interactions that constitute binding and therefore efficiently guide functional design. For instance, a model of a protein's predicted binding site can reveal favorable mutations at the protein-protein interface that strengthen the binding interaction through increase hydrogen bonding, thereby outcompeting other proteins. During my thesis project, I developed and implemented a series of computational modeling and design pipelines for generating better BMP-2 binders.

Protein folding, modeling, and design

From the experiments of Anfinsen, we know that protein sequence, or the length and order of the amino acid chain that makes up the primary structure of a protein, ultimately informs protein structure, or the 3D shape that the protein adopts [6] due to the minimization of energy during folding. While some proteins need assistance to fold into their native structure from molecular machines like chaperones, it is generally true that a protein with a given amino acid sequence will fold into one unique structure that represents the most stable, or lowest energy, conformation of the protein [7]. This phenomenon is often denoted as the protein folding problem in the field of protein engineering as it opens the possibilities for us to predict a protein's structure simply from its sequence. Indeed, many scientists have accepted the challenge of protein modeling and developed incredible computational tools capable of modeling proteins with atomic resolution from sequence information alone.

The opposite of the folding problem is the design problem, wherein we can imagine ourselves moving backward from a final, folded protein structure to the optimal sequence that encodes that structure. In nature, proteins adopt incredibly diverse shapes to do incredibly diverse functions. While evolution is responsible for the naturally occurring proteins that we observe, we can now employ the decades of knowledge we have gathered about protein biophysics and biochemistry to explore the infinite possible structures that proteins can take on, conceptualized as structure space. Sequence space is the corresponding infinite combinations of amino acids that inform such structures. De novo design—meaning protein sequence optimization that does not derive from evolutionary information—enables protein engineers to probe the vast uncharted structure and sequence spaces that remain untouched by evolution but hold incredible potential for the design of novel molecular machines, biomaterials, and therapeutic drugs and vaccines [8, 9, 10, 11].

The osteoinductive growth factor, BMP-2, is just one member of a family of bone morphogenetic proteins, many of which share nearly identical structures, save for a few non-conserved regions [3]. To control biologic delivery in a protein-rich biological healing environment, it is crucial that binders interact with BMP-2 and not its close relatives, some of which inhibit bone mineralization. In this project, I propose a pipeline for protein modeling and design augmented by machine learning to predict binding affinity to guide binder selectivity toward BMP-2 alone.

Rational design approach

Here, I will describe the computational approaches I took to efficiently obtain BMP-2 binders as well as the experimental methods to characterize them. In chapter one, I outline the modeling of existing affibody-BMP-2 interactions to modulate affibody affinity with mutagenesis. In chapter 2, I describe the steps I took to rationally design site-specific BMP-2 binders with novel topologies. Finally, chapter 3 covers the machine learning models that I explored in order to develop an algorithm for multi-state design of selective BMP-2 binders.

Both modeling and design of proteins and their intended interactions require the parameterization of biophysical and chemical forces acting on the constituent atoms of the proteins and their environment in order to accurately represent the natural biological system. I used Rosetta, a modular software suite which enables the generation of custom protein modeling and design pipelines [12], to reveal the structural basis of the BMP-2-affibody interaction as well as design novel interactions. To increase the success of novel designs, motifs or structural elements of a native binding interactions are extracted from crystal structures of natural protein-protein complexes and seeded or grafted into the backbone of a de novo scaffold, in a technique called motif-grafting [13]. This technique can be used to guide the site-specificity of a design, or the likelihood of designed binders to interact with one particular surface on a target protein.

Recently, protein designers have been hoping to move away from this technique, instead generating new computational pipelines that use the structure of the target protein alone to guide binder design, relying on highly accurate protein modeling and efficient sampling of sequence and structure space [14,15]. The advantage of these new methods is that design would no longer rely on the existence of relevant and accurate crystal structures. While promising, these pipelines have extremely low success rates and must be further improved to rival the efficacy of motif-grafting. Also at the forefront of the field is the development and implementation of machine learning to rapidly predict properties of proteins or protein-protein interactions, which can then be iterated with computational design for more robust exploration and exploitation of sequence and structure space [11, 16].

To computationally design site-specific and selective protein binders I use Rosetta, which employs physics- and heuristics-based calculations to assess the free energy of a protein structure. The field of protein engineering generates massive amounts of biophysical and statistical data, allowing researchers to leverage the power of machine learning for protein classification and prediction problems. To utilize data science for binder design, I developed a machine learning model that will predict the binding affinity between two proteins using Rosetta metrics as input. In the future, I will incorporate this ML model into an iterative sequence exploration and optimization program for multi-state design of binder proteins.

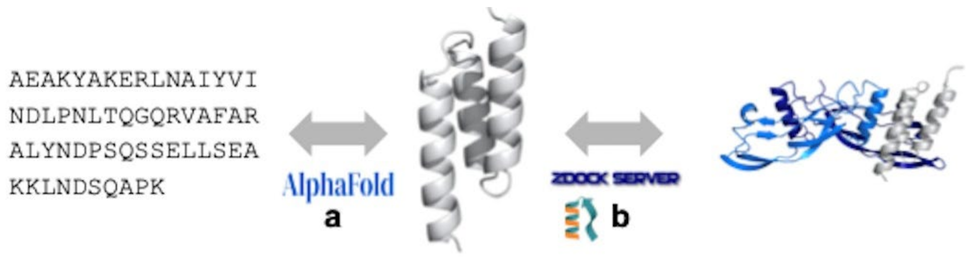


Figure 1. (a) AlphaFold predicts 3D structure from a protein sequence with state-of-the-art deep learning. (b) ZDOCK and Rosetta generate docked conformations of the affibody-BMP-2 complex. In order to design novel binders, we can implement the reverse, going from function to sequence.

Chapter 1: Computational Modeling of Affibody-BMP-2 Interactions

Introduction

Advances in computational protein structure prediction and docking, or the process of modeling protein-protein interactions (PPIs), have enabled researchers to predict the structural basis of binding with atomic level accuracy. Such high resolution is crucial as many PPIs depend upon the participation of particular atoms in intermolecular interactions. Hydrogen bonding, electrostatic interactions, and hydrophobic packing interactions help to stabilize the PPI, making the overall process of binding energetically favorable. Here I use computational modeling tools to investigate the binding interactions of affibodies against BMP-2 that are experimentally selected and characterized from a large library. Models of such PPIs offer a structural basis for further optimization and downstream characterization, such as the prediction of affibody mutants with different binding affinities. I hypothesized that affibodies would bind to one of the two receptor binding sites or epitopes on the BMP-2 surface. These epitopes have been characterized with X-ray crystallography and have been named the "wrist" and the "knuckle" to indicate their resemblance to the corresponding parts of hands, wherein the BMP-2 dimer is akin to a handshake (fig. 6).

My pipeline begins with structure prediction of affibodies, then docking of the PPI, then mutant prediction. While recent advances in deep learning have made structure prediction less challenging, docking continues to be a challenging modeling problem within computational structural biology. As a result, I have designed a pipeline that utilizes several checkpoints and iterations to refine docking models. Models of affibody-BMP-2 interactions confirm that affibodies prefer to bind to the "wrist," a helical binding epitope of BMP-2, where BMP-2 cofactors and type 2 extracellular receptors bind physiologically to begin the BMP-2 signal transduction pathway for bone mineralization. This docking model provides insight into a targetable binding site for the design of novel BMP-2 binders and atomistic detail for rational design.

Computational design of affibody mutants with affinity for BMP-2 in the low micromolar range, could offer more physiologically optimal release from a hydrogel into a bone fracture while minimizing off-target interactions. Binding interactions with serum proteins like fibronectin have been previously shown to accelerate BMP-2 release, causing uncontrolled temporal and spatial delivery and, as a result heterotopic, abnormal bone growth. Therefore, specificity in binding is crucial. In the future, I would endeavor to model affibody interactions with serum proteins like fibronectin and proteins intended to be co-delivered with BMP-2, such as FGF and GMCSF.

Methods

Computational

Public servers and the University of Oregon Talapas server are used to perform computational protocols. Rosetta licensed for the Hosseinzadeh lab is used to perform all RosettaScripts protocols. PyMOL is used for molecule visualization. The python pandas package is used for data analysis and matplotlib and seaborn are used for data visualization.

Structure prediction and preparation

Collaborators in the Hettiaratchi lab at University of Oregon obtained Sanger sequencing (Azenta) results of affibody vectors (Appendix 2) and translated them into amino acid sequences (Expasy). I input the affibody amino acid sequences into AlphaFold to predict the structure of the affibodies (as well as Robetta, however I used AF results for modeling). I stored the output 3D protein structures as PDB files. I then input PDB files into a RosettaScripts program running FastRelax to minimize the energy of predicted structures. FastRelax optimizes the protein structure through stochastic and iterative backbone and side chain packing while ramping up repulsive forces to model atomic packing, and generated PDB outputs.

Interface analysis of affibody-BMP-2 interaction

I input relaxed structures as PDB files into ZDOCK, an online server for docking which outputs a file with the top ten docking predictions. These predictions are relatively low refinement. They have not yet been characterized using the handpicked metrics of the Rosetta score calculation, a weighted value which approximates free energy based on physical and heuristic models. I completed initial visual inspection of the top ten predictions from ZDOCK using PyMOL. In order to probe specific biophysical features at the interfaces of these predicted PPIs to explore the metrics which most influence binding affinity, I wrote an interface analysis RosettaScripts program (Appendix 1) to calculate these features and output them for each prediction in an array structure. I analyzed the computational feature data in Jupyter Notebooks.

Ensemble analysis

Protein-protein interactions are dynamic in physiological circumstances, allowing for small movements of atoms and even the macromolecules themselves, in relation to one another. Therefore, modeling a protein's dynamics may improve interface analysis of a binding interaction. I generated a collection of decoys, or PPI conformations, that would model the diversity of these structures. Ensemble generation and metrics on each decoy were performed in another Rosetta program that utilized an algorithmic "mover" in RosettaScripts called BackrubDDMover. This mover makes small movements in the backbone to model the movements that would occur physiologically upon binding of the affibody to BMP-2. I analyzed the ensemble of decoys using interface analysis metrics (Appendix 1).

RosettaDock

I performed refined docking to more precisely model the positions of atoms in the affibody-BMP-2 complex using the RosettaDock application. This program samples a large number of decoys and assesses the thermodynamic parameters associated with each. I generated 2000 decoys for protein dynamic analysis. I analyzed the biophysical data output for all decoys using Pandas in Jupyter Notebooks. I then selected one low energy decoy as the final, refined PPI model.

Computational site-saturation mutagenesis

I performed computational site-saturation mutagenesis (SSM) in order to investigate the individual contributions and importances to binding of each interfacial residue of the affibody. This approach selects all residues of the affibody that occur within 8 angstroms of BMP-2, mutates them to all twenty amino acids, and records the delta delta G ($\Delta\Delta G$) or free folding energy of each point or single mutation. $\Delta\Delta G$ is a proxy for binding energy in the case of protein complexes. I input each refined docking prediction from RosettaDock into a RosettaScripts program which performs computational SSM using an algorithmic mover called GreedyOptMutationMover. The recorded ΔΔG data is stored in a table where the rows are the residue numbers and the columns are the amino acids to which that residue is mutated. The table values store the ΔΔG for each single mutant. I visualized the tabular output with a seaborn heatmap in Python (Appendix 1).

Mutant generation

I generated single mutants from computational SSM by visualizing the wild type affibody in PyMOL and using the Wizard feature to mutagenize the original residue to the mutant residue. This is done for each single mutant. To minimize the energy of these manually generated mutants, I then used the RosettaScripts platform to perform a relax on the PDBs. These relaxed point mutants are then ready to be computationally assessed for BMP-2-binding ability.

Point Mutant Docking

Point mutants are docked using RosettaDock with the same protocol used for the wild type affibody-BMP-2 PPIs and subsequent data analysis in Python is performed to assess the docking using 2,000 decoys (Appendix 1).

Experimental

I selected six single mutants of one of the modeled affibodies to be experimentally characterized in order to validate the accuracy and efficacy of the computational pipeline for generating affibody mutants.

Site-directed mutagenesis primer design

For site-directed mutagenesis to obtain G3C1 affibody mutants, I manually designed forward and reverse primers to have melting temperatures (Tm) within 1.5ºC of each other. The primer pairs have overlapping regions less than half the length of the full primer length (colored in red and blue) which include the mutated residue codon (lowercase). The overlapping and non-overlapping primer regions were designed to have Tm within 5-10ºC of each other. I analyzed heterodimer Delta G of each primer pair and checked overlap length using the IDT Codon Analyzer Tool. I translated the forward primer (red and black) from a nucleotide sequence into an amino acid sequence using Expasy Translate to confirm the mutation. I performed NCBI nucleotide Blast for each primer, where the primer was the query and the pCT40 vector including G3C1 affibody sequence was the subject.

V28W
5' CAGAGAtggGCATTCGCGCGGGCACTGTATAACGAC ; Tm=68.8 ºC, length=36
5' CGAATGCccaTCTCTGACCCTGGGTCAGGTTCGG ; Tm=69.1 ºC, length=34
heterodimer Delta G: -31.57 kcal/mole, Base Pairs: 16
Expasy translated forward primer amino acid sequence: QRWAFARALYND
NCBI Nucleotide Blast: forward primer alignment: 33/36, reverse primer alignment: 31/34

V28T
5' CAGAGAaccGCATTCGCGCGGGCACTGTATAACGAC ; Tm=68.8 ºC, length=36
5' CGAATGCggtTCTCTGACCCTGGGTCAGGTTCGG ; Tm=69.2 ºC, length=34
heterodimer Delta G: -31.97 kcal/mole Base Pairs: 16
Expasy translated forward primer amino acid sequence: QRTAFARALYND

NCBI Nucleotide Blast: forward primer alignment: 34/36, reverse primer alignment: 32/34

V28D
5' CAGAGAgacGCATTCGCGCGGGCACTGTATAACGAC ; Tm=68.5 ºC, len=36
5' CGAATGCgtcTCTCTGACCCTGGGTCAGGTTCGG ; Tm=, len=68.8 ºC, len=34
heterodimer Delta G: -30.13 kcal/mole Base Pairs: 16
Expasy translated forward primer amino acid sequence: QRDAFARALYND
NCBI Nucleotide Blast: forward primer alignment: 35/36, reverse primer alignment: 33/34

A33Q
5' GCGCGGcagCTGTATAACGACCCGTCCCAGAGCTCTG ; Tm=71.6 ºC, len=37
5' ATACAGctgCCGCGCGAATGCGACTCTCTGACCCTGGGTC ; Tm=72.4 ºC, len=40, heterodimer Delta G: -33.73 kcal/mole Base Pairs: 15
Expasy translated forward primer amino acid sequence: ARQLYNDPSQSS
NCBI Nucleotide Blast: forward primer alignment: 34/37, reverse primer alignment: 37/40

A33L
5' GCGCGGttgCTGTATAACGACCCGTCCCAGAGCTC ; Tm=70.1 ºC, len=35
5' ATACAGcaaCCGCGCGAATGCGACTCTCTGACCCTGGG ; Tm=71.2 ºC, len=38,
heterodimer Delta G: -32.28 kcal/mole Base Pairs: 15
Expasy translated forward primer amino acid sequence: ARLLYNDPSQS
NCBI Nucleotide Blast: forward primer alignment: 32/35, reverse primer alignment: 35/38

A33Y
5' GCGCGGtatCTGTATAACGACCCGTCCCAGAG ; Tm=66.6 ºC, len=32
5' ATACAGataCCGCGCGAATGCGACTCTCTGACCCTG ; Tm=67.7 ºC, len=36
heterodimer Delta G: -29.25 kcal/mole Base Pairs: 15
Expasy translated forward primer amino acid sequence: ARYLYNDPSQ
NCBI Nucleotide Blast: forward primer alignment: 29/32, reverse primer alignment: 33/36

Vector and vector linearization

The pCT40 vector from Dane Wittrup (Appendix 2) is used here for characterization with yeast surface display and flow cytometry. Vector is extracted from one of the experimentally selected affibody cultures using the yeast plasmid extraction kit from Zymoprep. The vector is then linearized and amplified using primers on the outside of the affibody sequence region and PCR. The PCR product concentration is measured using Nanodrop spectrophotometry. I obtained linearized vector which excludes the affibody sequence to enable yeast homologous recombination. I linearized vectors using forward and reverse primers outside the affibody sequence. I followed the Takara Bio HiFi PCR Premix Quick Protocol for linearization PCR (Thermocycler). I digested PCR product using Dpn1 restriction enzyme according to the Dpn1 digest protocol from NEB. I ran a 1% agarose gel to confirm the linearized plasmid size.

G-Block mutant sequence design

For homologous recombination to obtain G3C1 affibody mutants, I designed seven double stranded DNA segments–one positive control and six mutant sequences–intended for transformation into competent yeast along with linearized pCT40 vector, which excludes the affibody sequence. Yeast perform homologous recombination to join dsDNA inserts (g-blocks) to pCT40 using overlapping regions of the vector from 17-30 base pairs long. I used the IDT Codon Optimization Tool for Saccharomyces Cerevisiae to translate amino acid sequences into nucleotide sequences. We ordered final g-block sequences from IDT. Sequences and g-block details in Appendix 2.

Homologous recombination and yeast transformation

Yeast cells possess the ability to perform homologous recombination *in vivo*, potentially increasing the efficiency of *in vitro* experimental manipulation to insert the mutant sequence into the vector for expression into EBY100 chemically competent yeast.

Linearized vector and mutant affibody g-blocks (IDT) are transformed into yeast using the frozen EZ yeast transformation kit (Zymoprep). Yeast are recovered for half an hour in 5ml YPD (Hackel) and then plated in a 2000X dilution of PBSA onto SDCAA (Hackel).

Liquid culture plasmid extraction and Sanger sequencing

Half of a transformed colony grown on SDCAA is inoculated into 5ml SDCAA liquid media and grown for 16-18 hours at 30˚C and 250 rpm. 1.5mL of this culture is used to extract yeast plasmid according to Miniprep kit instructions (Zymoprep) and eluted in 15ul DNA elution buffer. A sample of 5-7 ul of pure plasmid at at least 50ng/ul is submitted to same-day Genewhiz Sanger sequencing using the Azenta platform and primed off-site using T7 primer. Sequences are analyzed first in CLC Sequencer and then compared to target sequences using NCBI Nucleotide Blast and Expasy Translate public online servers.

Flow cytometry on yeast surface display

Samples for flow cytometry are prepared according to Benjamin Hackel et al. protocols used by the Hettiaratchi lab. One colony from the mutant affibody sample plated on SDCAA is used to inoculate 15ml SDCAA with ampicillin (100 mg/ml) and ciprofloxicin (1 mg/ml) and cultures are grown in a 125ml baffled flask at 30˚C, 250rpm for 18 hours. Cultures are then centrifuged at 5000g for 3 minutes and the pellet is resuspended in 1ml of SGCAA and added to 15 ml SGCAA with ampicillin and ciprofloxicin and incubated in 125ml baffled flask at 30ºC, 250rpm for 24 hours.

Cultures should be cloudy. Cell densities are measured using a Countess machine and In Vitrogen Countess slides (3ul from cell culture added to 597ul PBSA; 10ul added to slide; cell concentration recorded as two orders of magnitude greater than the Countess reading provided). One million cells are used for each sample for flow cytometry. Flow cytometry (FC) experiments should be conducted with a positive control, negative control (EBY100 grown in YPD), and experimental sample. For each sample (controls and treatments), FC is run on wells with cells only, secondary antibody solution and cells, secondary antibody solution and 2.5 ul anti-C-myc antibody and cells, secondary antibody solution and 2.5ul anti-HA antibody and cells, secondary antibody solution and biotinylted BMP-2 and cells, secondary antibody solution and biotinylated BMP-2 and anti-C-myc antibody and cells, or secondary antibody solution and biotinylated BMP-2 and anti-HA antibody and cells (7 wells per sample). Secondary solution contains Alexa Fluoraphore 647 and Alexa Fluoraphore 488 (333nM).

Flow cytometry experiments are conducted on the BD6 Acuri automated flowcytometer (FC). Limits of 10,000 events, 150ul, and 3 minutes are set under manual controls. The medium flow setting is used for FC experiments. After primary antibodies and/or BMP-2 and cells are mixed in Eppendorf tubes, samples are shake incubated in 4ºC for 1 hour. Samples are then washed with 500ul PBSA and resuspended in 50ul secondary antibody solution (except cells only samples which are resuspended in 200ul PBSA and set aside). Samples are incubated in dark conditions and 4ºC for 15 minutes before being washed twice with with 500 and then 800ul PBSA and resuspended in 200ul PBSA for transfer to a chilled, flat-bottom 96-well plate.

Results

My computational pipeline to model protein-protein interactions between BMP-2 and several affibodies revealed a consistent affibody preference for the wrist epitope of BMP-2. Prior to modeling, our collaborators obtained Sanger sequence data of seven affibodies selected from a large library during MACS to obtain functional protein binders. Affibody library diversity is produced by 17 variable residues out of the total 58 residues of the affibody scaffold (fig. 2a).

With unique affibody sequences as input, I predicted structures of affibodies using Robetta and AlphaFold, both of which generate confidence plots upon prediction. Confidence plots from AlphaFold show that five out of seven affibodies are predicted with high confidence, above 70% pLDDT or predicted local difference distance test score, while the other two affibodies are less confident overall (fig. 2b). The predicted 3D structures of affibodies correspond with these results, showing that five of the affibodies are regularly structured in three well-defined alpha helices (fig. 2c) while two of the affibodies are less tightly packed (fig. 2d), in addition to being less confidently predicted.

I then did a low-refinement docking in the publicly accessible server, ZDOCK. ZDOCK outputs its top ten structures for each PPI (fig. 3), which I then minimized using FastRelax in Rosetta, a step to computationally optimize the complex structure using small torsion movements. This outputs a “score" file where the lowest scoring complex out of ten, whose score is based on the Rosetta energy score function, is chosen as the lowest energy and therefore most likely to occur. In addition, I analyzed the affibody-BMP-2 interface using other biophysical metrics such as $\Delta\Delta G$ and SASA.

Data analysis in which I compared the metrics of different predictions against one another led to a deeper understanding of the likely preferences of the affibodies toward certain sites of BMP-2 due to both geometric and chemical interaction predictions and calculations.

Ultimately, score (a Rosetta weighted energy term in REU, Rosetta energy units), $\Delta\Delta G$ (or change in free folding energy), shape complementarity (or geometric surface fit), and SASA (or solvent accessibility) led me to select three out of ten of the ZDOCK outputs to further characterize (fig. 4).

Data analysis on ensembles rather than singular protein interactions offers a more holistic view of the PPI and its dynamics. In order to model the effect of dynamic movements on the favorability of each of the three chosen protein-protein interactions, I generated ensembles of complex decoys, or slightly moved conformations of the complex between an affibody and BMP-2, using the BackrubDD mover of Rosetta. For all decoys, I plotted the distribution of the score value. Data analysis of the RosettaDock output demonstrates a positive correlation between root mean square (RMS) and thermodynamic features of the PPI, essentially showing that more energetically favorable PPI conformations have a particular structural distinction from the starting input structure. A funnel shape in the graph of RMS versus interface score (I_sc) shows that the PPI falls into an energetic minima for a folded and bound structure. The lowest energy, and therefore most favorable, decoy is selected from RosettaDock as the most refined conformation. The violin plots corresponding to each complex demonstrate that two out of the three complexes were more favorable, having a lower score distribution (fig. 5).

Not only were they more favorable, these two complexes both showed that the affibody docks to BMP-2 in a well-defined pocket of the BMP-2 protein known as the wrist. This pocket or epitope is the location of BMP-2 binding to one of its two receptors. The other binding epitope on the surface of the BMP-2 protein that participates in receptor-binding is known as the knuckle and consists primarily of a twisted beta sheet (fig. 6). Seven affibodies structures were predicted and docked to BMP-2 using ZDOCK and RosettaDock, demonstrating the preference of the affibody scaffold, a three helical bundle topology, for the wrist epitope. This preference is likely due to the well-matches shapes of the secondary structures of the BMP-2 wrist and the affibody scaffold, whose surface exposed side chains may make hydrophobic, as in the case of leucine, or hydrogen bonding interactions, as in the case of aspartic acid.

In order to explore the modulation of affibody affinity for BMP-2, I used computational site saturation mutagenesis (SSM) to compute the change in $\Delta\Delta G$ as a result of the *in silico* mutation of each interface residue of the affibody to all 19 other amino acids. Experimental methods such as alanine scanning, mutating residues to alanine, or deep mutational scanning (DMS), in which site saturation mutagenesis is followed by deep sequencing, are costly and labor intensive in comparison to computational SSM. In order to perform computational mutagenesis, I used Rosetta's GreedyOptMutationMover and decided to focus my data analysis on point mutations which increased $\Delta\Delta G$, from more favorable, negative values, to less favorable, positive values (fig. 7). I binned these point mutations into low, medium, and high change and selected 2-3 PMs from each bin. I selected point mutants for each affibody based on both the heatmap visualizations of change in $\Delta\Delta G$ (fig. 7) and visual inspection of side chain contact at the interface with PyMOL. These mutants are predicted to resukt in a less favorable ΔΔG, suggesting that the binding interaction between the mutant and BMP-2 is weaker than the wild-type interaction. A weaker binding interaction that still results in binding at the computationally predicted site is here hypothesized to generate binders which have micromolar affinity and an increased release rate over a physiological relevant timescale (months) in which bone healing occurs. Point mutants that result in less favorable ΔΔG are then binned according to change in ΔΔG and several point mutants are selected for computational assessment prior to experimental characterization.

To computationally generate point mutants, I used the mutagenesis Wizard tool in PyMOL and relaxed subsequent point mutants with Rosetta's FastRelax protocol to energetically minimize the structures before docking them to BMP-2 with RosettaDock. Point mutants that were predicted to dock to the BMP-2 wrist epitope demonstrated a characteristic funneling during docking toward 0 rms from the input structure and the lowest I_sc or interface score. I selected six point mutants of the G3C1 affibody to characterize experimentally (fig. 8). I generated point mutant nucleotide sequences by modifying Sanger sequencing results of the original affibody (Appendix 2) to incorporate the point mutation optimized for yeast expression. I first attempted to experimentally mutagenize the G3C1-affibody sequence contained within the pCT40 vector using site directed mutagenesis with non-overlapping and partially overlapping primer sequences. The low efficiency of this method led me to try using yeast homologous recombination in order to experimentally generate mutants instead. I expressed point mutants in competent yeast through transformation and homologous recombination, confirming mutant expression using clonal PCR or overnight amplification and Sanger sequencing. Results of the initial flow cytometry analysis of these mutants have been inconclusive due to the growth of negative controls and the lack of induction of yeast to express mutant affibodies. Future reports will detail the results of more extensive flow cytometry analysis and binding affinity measurements.

Conclusion

Modeling affibody conformation and docking has provided structural insight into the interaction between binding affibodies and BMP-2. Affibodies are predicted to bind the wrist epitope of BMP-2, a function site on the protein. Computational SSM further probes the specific residues that contribute strongly to binding and can thus modify the binding dynamics if mutated. I have selected and experimentally characterized such mutants with the goal of weakening the original affinity of the binding interaction for more effective and physiologically relevant BMP-2 release into regenerating bone tissue. As computational simulations of energetic force fields and heuristic protein features improve, models of protein-protein interactions become a more enticing tool for protein engineering. The relative simplicity of the modeling pipeline described here allows the tools and models to be increasingly accessible. Ultimately, my PPI models of affibodies and BMP-2, as well as the modeling pipeline itself, will continue to direct protein engineering efforts in BMP-2 delivery to augment bone healing.

Figures

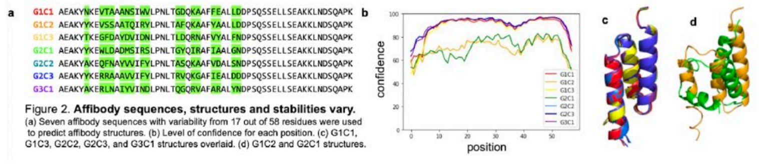


Figure 2. **Affibody sequences, structures and stabilities vary.** (a) Seven affibody sequences with variability from 17 out of 58 residues were used to predict affibody structures. (b) Level of confidence for each position. (c) G1C1, G1C3, G2C2, G2C3, and G3C1 structures overlaid. (d) G1C2 and G2C1 structures.

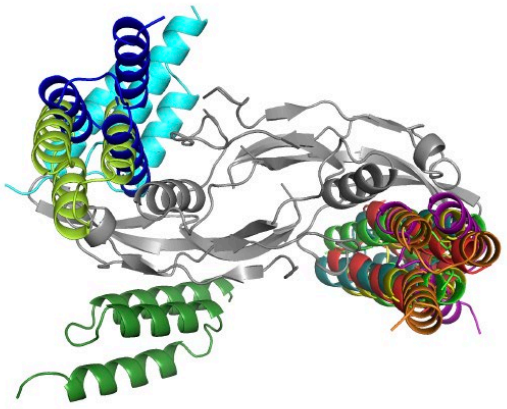


Figure 3. Top ten ZDOCK predictions for the G3C1 affibody (in ten colors) and BMP-2 (grey).

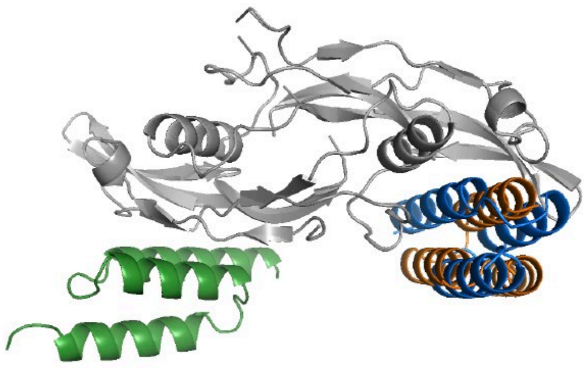


Figure 4. Lowest energy binding conformations between the G3C1 affibody (green, blue, orange) and BMP-2 (grey).

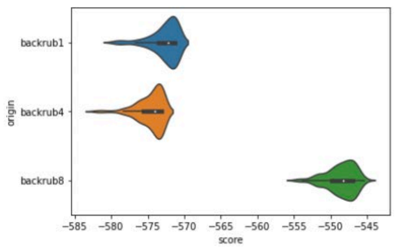


Figure 5. PPI ensembles for each docked conformation in figure 4 were generated to model dynamic molecular movements.

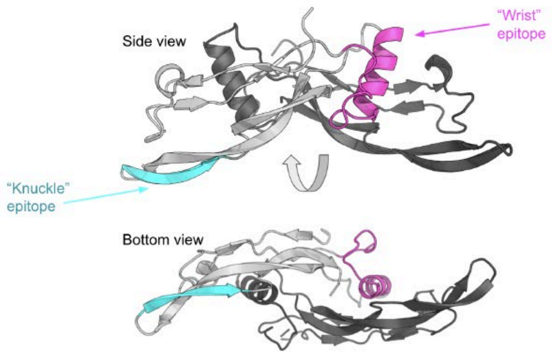


Figure 6. BMP-2 (grey) binding epitopes for two receptors are called the wrist (pink) and the knuckle (cyan).

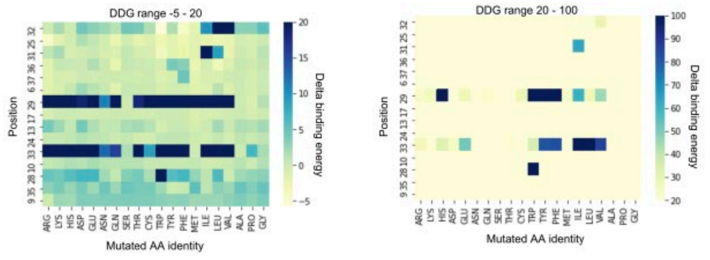


Figure 7. Change in ΔΔG for each point mutant of the G3C1 affibody interface with BMP-2.

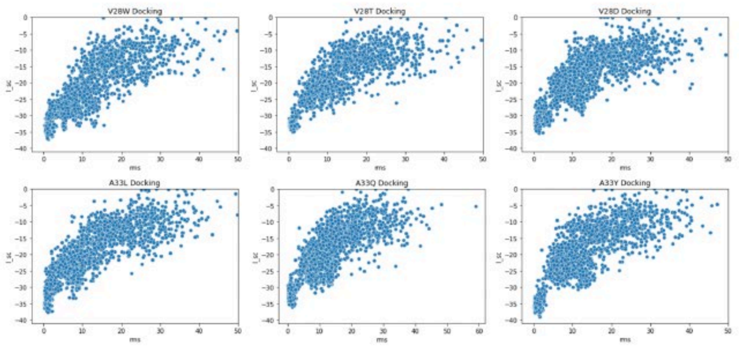


Figure 8. Docking results from six point mutants of the G3C1 affibody. Rms is root mean square, a measure of global structural difference, and I_sc is interface score, or the Rosetta score at the protein-protein interface.

Chapter 2: Rational Design of Novel BMP-2 Protein Binders

Introduction

In physiological BMP-2 signaling, the growth factor simultaneously engages with two receptors on the surface of osteoblast progenitor cells, which triggers a signal transduction cascade resulting in the differentiation of the precursor cell to the osteogenic cell. BMP-2 binds its two receptors using two unique epitopes: the wrist and the knuckle. Solved X-ray crystal structures of BMP-2-receptor interactions offer researchers a structural understanding of BMP-2 activity in development and bone regeneration. Structural characterization can be efficiently translated into clinically relevant work on therapeutics and biologic delivery mechanisms. On the other hand, experimental selection pipelines which use magnetic bead sorting or fluorescence activated cell sorting to select BMP-2 binders require downstream optimization and characterization. Such pipelines fail to perform site-specific selection resulting in a lack of structural information about binding, such as the interacting surfaces, residues, and orientations.

Protein structure prediction of experimentally selected affibodies indicates highly conserved scaffold structure. Due to the conserved protein topology of the three-helical affibody library, we hypothesized that binders would prefer to bind one specific site regardless of the availability of other sites or epitopes on the target protein. For these reasons, rational binder design using computational methodologies offers advantageous control over the selectivity and specificity of the binder to a particular target site and access to more than one site on the target protein using a variety of sized and shaped scaffolds for design. In this project I perform design of novel binders to two prominent epitopes for native receptor-binding to BMP-2 with the computationally intensive but effective algorithms and force field approximations of Rosetta. Low throughput and high throughput pipelines for novel binder design are described here. I then outline the pipeline to experimentally characterize designs using yeast surface display and flow cytometry analysis or fluorescent activated cell sorting (FACS) and deep Illumina sequencing. In the future, structure-guided design of BMP-2 binders may be extended to even more surfaces on the target protein for wider functionality and deliverability.

Methods

Computational

Public servers and the University of Oregon Talapas server are used to perform computational protocols. Rosetta licensed for the Hosseinzadeh lab is used to perform all RosettaScripts protocols.

Scaffold library generation

Stable, mini-protein scaffold libraries from Rocklin (2017), Strauss (2021), and truncated scaffolds from Brunette (2019) are collected and used for motif grafting. Truncation of helical repeat proteins from Brunette 2019 was accomplished by manual selection of highly resolved scaffolds designed without disulfide bridges and visualization in PyMOL to truncate the scaffold after the fourth helical repeat. The surface was then redesigned to stabilize the newly solvent-exposed region of the scaffold with Rosetta, written up in a script called designsurface.xml (Appendix 1). Scaffolds range from 38-98 amino acids in length.

Motif grafting

Native binding motifs were extracted from crystal structures of BMP-2 and RGMC (PDB ID: 4ui1) or ALK1 (PDB ID: 6sf2) to seed design at the wrist epitope and knuckle epitope of BMP-2. Motif grafting was then performed in RosettaScripts with the MotifGraft Mover. Motif_graft.xml is contained in Appendix 1.

Graft diversification

Outputs from wrist motif grafting were diversified using BackrubDD Mover for the high throughout pipeline. This step was omitted for low throughput design. In the high throughput pipeline, over 75,000 designs were generated after this step, in which each successful graft was diversified approximately 20-fold. Backrub_grafts.xml is contained in Appendix 1.

Sequence optimization and interface minimization

The FastDesign Mover in RosettaScripts was used to optimize sequences for the BMP-2 interface and for hydrophobic packing in a script called optimize_sequence.xml (Appendix 1). In the high throughput pipeline, special care was taken to prevent the redesign of loops, prolines, glycines, and core residues. The interface was then minimized with the FastRelax mover to improve packing and rotamer positioning.

Filtering

Designs were then filtered in the low throughput pipeline using several metrics including score, ddg, sasa, packstat, shape complementarity, interface holes, and buried unsatisfied polar atoms. For LTP: score < 0, ddg_norepack < 0, shape_complement > 0.6, packstat > 0.6, interface_holes < 1.5, and buried_unsat < 1. The high throughput pipeline was similarly filtered using metrics inspired by those used in the thesis work of Brian Coventry (2021). For HTP: score < 0, cms > 450, ddg_norep < -30, mismatch_probability < 0.5, contact > 100, and t_sap_score < 50. The low throughput pipeline yielded nine promising designs that were subsequently docked using RosettaDock with the same protocol as described in chapter 1 and narrowed down to six designed for experimental characterization based on visual inspection of cavities, hydrophobic interactions, and hydrogen bonding networks using PyMOL. For the high throughput pipeline, 1017 designs passed the HTP filters.

Experimental

Six low throughput designs were selected to be experimentally characterized in order to validate the accuracy and efficacy of the computational pipeline for generating novel BMP-2 binders to the wrist and knuckle epitopes.

Library primer design

The high throughput pipeline yielded more than 1000 designs to test, generating the necessity to design and synthesize a library for yeast transformation. Primers for flow cell attachment and synthesis reaction sequencing for Illumina were designed and ordered through IDT. G-blocks for design sequences were ordered through Twist.

Vector

The pCT40 vector from K. Dane Wittrup (Appendix 2) is used here for characterization with yeast surface display and flow cytometry or FACS. Vector is extracted from one of the experimentally selected affibody cultures using the yeast plasmid extraction kit from Zymoprep. The vector is then linearized and amplified using primers on the outside of the design sequence region and PCR. The PCR product concentration is measured using Nanodrop spectrophotometry.

Homologous recombination and yeast transformation

Same protocol as described in Chapter 1 methods.

Flow cytometry and FACS for yeast surface display

Same protocol as described in Chapter 1 methods.

Results

One primary concern in the design of protein binders to native, or naturally occurring, targets such as BMP-2 is non-specific or off-target binding, which may cause aberrant signaling and other off-target effects. To prevent non-specific interactions, I began the design of site-specific BMP-2 binders by identifying structural elements that have been experimentally validated through X-ray crystallography and are known to interact with BMP-2 in endogenous human protein signaling pathways. Since these structural elements, often 4-14 amino acids in length, are known to directly interface with BMP-2 in nature, they can be used as seeds to generate novel binders. These structural elements are called motifs and the process of generating whole protein binders around them is called *motif grafting* (fig. 9). The native binding interaction between BMP-2 and its cofactor RGMC, also known as hemojuvelin, provides a motif to guide the *de novo* design of protein binders with specificity toward the wrist epitope of BMP-2 (fig. 10). The wrist epitope consists of an alpha helix and a loop, both of which interact with the interfacial alpha helices of RGMC, as demonstrated in the crystal structure of the BMP-2-RGMC complex (PDB ID: 4ui1). Alpha helical-alpha helical interactions such as this one are often made favorable by both hydrophobic interactions. I chose to extract a binding motif from RGMC due to the rigidity and high predictability of the interfacial structural element, here, an alpha helix. The wrist epitope of BMP-2 binds to other proteins, like BMP receptors, but these interactions contain several loop regions, which are less predictable and relatively harder to design with than alpha helices.

In order to design binders that are site-specific to the knuckle epitope of BMP-2, I extracted a structural binding motif from a receptor that uses a beta sheet to make a specific interaction with the beta sheet of the knuckle (PDB ID: 6sf2). Such beta sheet-beta sheet interactions are challenging because they rely on hydrogen bonding (fig. 11). Hydrogen bonds use polar contacts, atoms that can either accept or donate a proton to maintain this relatively strong non-covalent interaction. In fact, hydrogen bonds between water molecules, which are polar molecules, are responsible for the surface tension we observe in nearly spherical droplets of water and the high heat required to boil a pot of water, which breaks these bonds to release individual water molecules as gas particles. In the case of the BMP-2 knuckle epitope, hydrogen bonding between two beta sheets requires the sheets to be parallel, which is complicated by the twisted conformation of the knuckle's beta sheet. Therefore, guiding design with a native-binding beta sheet motif increases the likelihood of maintaining that parallel interaction geometry. My colleague is also working on the design of BMP-2 knuckle binders using a beta-sheet extension pipeline developed by Sahtoe et al (2021). This program builds out a beta sheet binding element by analyzing the beta sheet of the knuckle epitope and generating a novel motif to guide specific binder design.

The binding motif alone is not stable enough to deliver BMP-2 to a bone fracture. Like intermolecular interactions interactions, favorable intramolecular contacts may stabilize a polypeptide, allowing it to maintain a stable 3D structure. Additionally, larger and more complex proteins than these motif peptides may increase the specificity and strength of the interaction by making favorable contacts with other BMP-2 surfaces near the binding epitopes. For these reasons, I inserted my extracted binding motifs into *de novo* protein scaffolds. *De novo*, translating to "from scratch," protein engineering aims to develop proteins with little to no sequence homology to natively occurring proteins for protein applications in biotechnology, biopharmaceuticals and biomaterials, to name a few. One category of *de novo* protein engineering that has expanded in recent years is the field of mini-protein design. Mini-proteins are small proteins, often ranging from 30-70 amino acids in length, which can, depending on their primary sequence, adopt various conformations with secondary structure elements like alpha helices, beta sheets, loops, and turns (fig. 12). The various shapes, or topologies, that are created by the combination of these secondary structure elements diversify the interactions these proteins can make and therefore the applications for which they can be used. This phenomenon is encapsulated in the form-function relationship of molecules, which states that a molecule's 3D structure informs its physiological function. Nature has generated a beautiful diversity of protein shapes and sizes to match each unique function that proteins have in and around cells. However, we can imagine even more shapes, which in turn, may produce novel functions ranging from biocatalysis to biomechanical signaling.

To design proteins which may stabilize the binding motifs extracted from native interactions with BMP-2, I computationally inserted the motifs into highly stable, *de novo* mini-protein scaffolds. Motif grafting in Rosetta allows the user to apply certain constraints on the insertion of such elements which may be appropriate for a given application. Here, I applied constraints to keep the motif as intact as possible while also changing the core of the scaffold minimally. These grafting constraints ensured that the motif would maintain enough structural integrity to remain site-specific while the scaffold would retain the intramolecular contacts that stabilize it in a highly predictable conformation (fig. 13). I then submitted the completed grafts to Rosetta's FastDesign program, which optimized the sequence for interactions with BMP-2 while, again, constraining design to maintain the original structural integrity of the motif and the scaffold. Improvements upon this binder design pipeline in the second round of design diversified and expanded the library of grafts to be optimized with FastDesign by generating 20-fold ensembles of the grafts through backrub (BackrubDD mover).

I then calculated a number of computational interface metrics that are shown to be important for optimal binding. The designs with best interface metrics values passed my metric filters (fig. 14) and were docked to analyze binding conformation predictability (fig. 15). After round one, six designs were selected to characterize, three designed to BMP-2 knuckle and three designed to the BMP-2 wrist. After round two, which included graft diversification and modifications to sequence optimization, 1017 designs passed my filters. Final *de novo* designed binders are predicted to bind to the wrist and knuckle epitopes of BMP-2 depending upon which motif (RGMC or ALP-I) was used in motif grafting (fig. 16).

To experimentally characterize binders, I have prepared designs from both rounds in a library, totaling 1023 designs. I will transformed this library of g-blocks sequences with linearized vector into competent yeast and analyze induction, or the rate of expressing the designs on the yeast surface, and binding to BMP-2 using fluorescent activated cell sorting or FACS (SH800). I plan to sort the yeast into high, moderate, and low affinity gates, extract the recombined vectors from these yeast, and amplify the sequences with enrichment PCR. I will then submit PCR products to Illumina sequencing at the G3CF core facility at University of Oregon in order to deep sequence each gate. I will analyze Illumina results to assess the success of my design library. Experimental characterization will not only provide information about which designed binders are successful at binding to BMP-2, it will generate qualitative data regarding the strength of binding which may be iteratively incorporated into the design pipeline using interface metric analysis data from Rosetta.

Conclusion

To generate site-specific BMP-2 binders, I extracted native-binding motifs and grafted them into stable mini-protein scaffolds, optimized grafts for BMP-2 binding, and finally filtered resultant designs to obtain the most successful designs to experimentally characterize. After two rounds of design, I expressed 1023 of my designs in yeast and performed yeast surface display experiments to separate high, moderate, and low affinity binders. Computational modeling of protein-protein interactions thus enabled me to factor structural insights into my design of site-specific binders. I will further characterize the affinity, stability, and structure of my novel, high, moderate, and low affinity BMP-2 binders to experimentally confirm the success of an efficient computational pipeline for site-specific protein binder design.

Figures

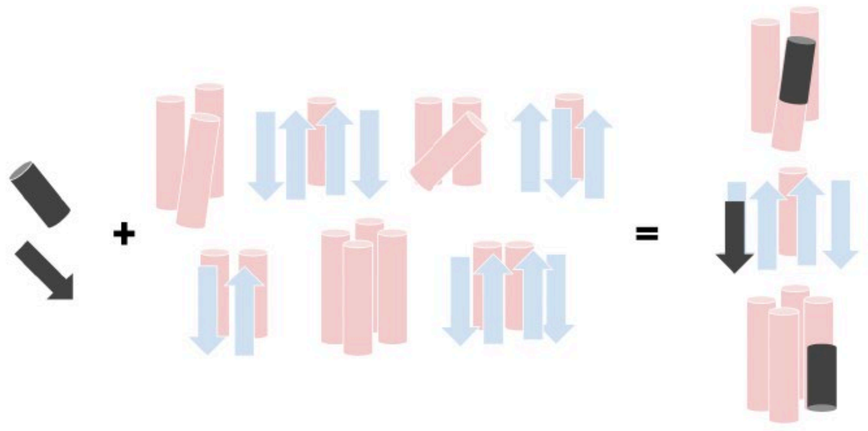


Figure 9. Schematic of motif grafting. Motifs are colored in dark grey while alpha helices (red) and beta sheets (blue) constitute the diverse topologies of the scaffolds. Grafted results are imperfectly aligned to represent the slight flexibility of motif grafting constraints into stable scaffolds.

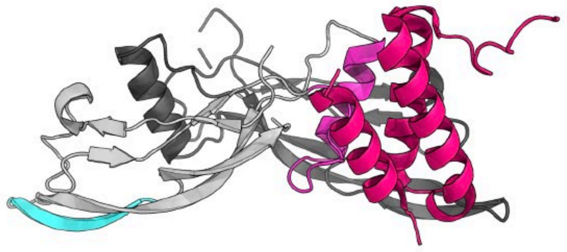


Figure 10. RGMC (fuchsia) bound to the wrist epitope of BMP-2 (magenta). PDB ID: 4ui1. Here, the binding interaction is helix-helix. Motif sequence: LAFHSAVHG.

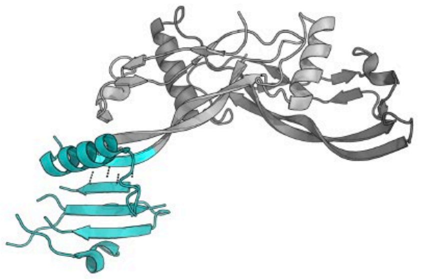


Figure 11. Act-I receptor extracellular domain bound to the knuckle epitope of BMP-2. The interaction is colored in cyan and hydrogen bonds across the protein-protein interface are indicated by dashed black lines. PDB ID: 6sf2. Here, the binding interaction is sheet-sheet. Motif sequence: SNIVRSFS.

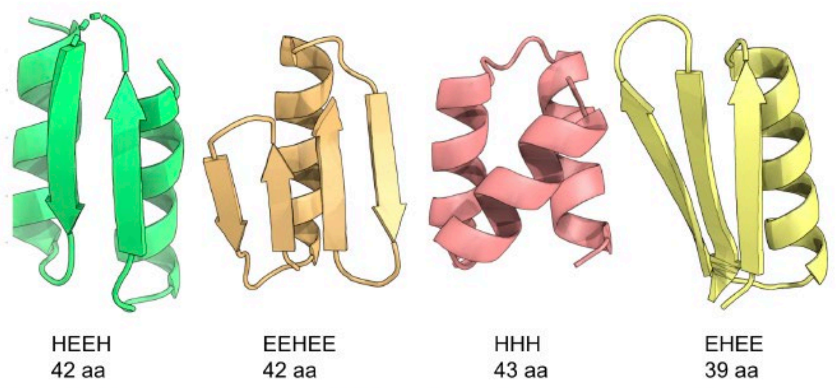
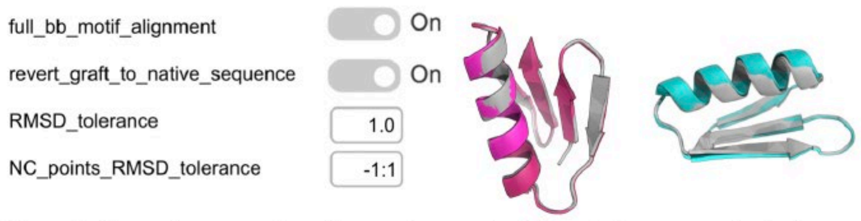


Figure 12. Diverse mini-protein scaffold topologies offer various possible binding interactions through motif grafting.



full_bb_motif_alignment On
revert_graft_to_native_sequence On
RMSD_tolerance 1.0
NC_points_RMSD_tolerance -1:1

Figure 13. Constraints on motif grafting produce predictable grafted structure using both RGMC (magenta) and ALP-1 (cyan) motifs. Grafted structures are overlaid with ungrafted scaffold structures in grey.

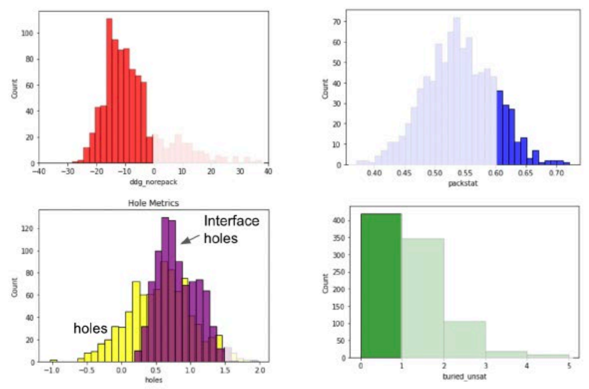


Figure 14. Filtering for the low throughput pipeline. Only the shaded areas of the graphs are allowed to pass through the filter.

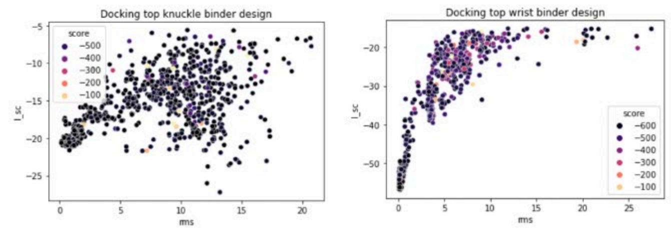


Figure 15. Docking results from the first round of design. The top scoring design using the wrist-binding RGMC motif more highly predictable than the top scoring design using knuckle-binding ALP-I motif.

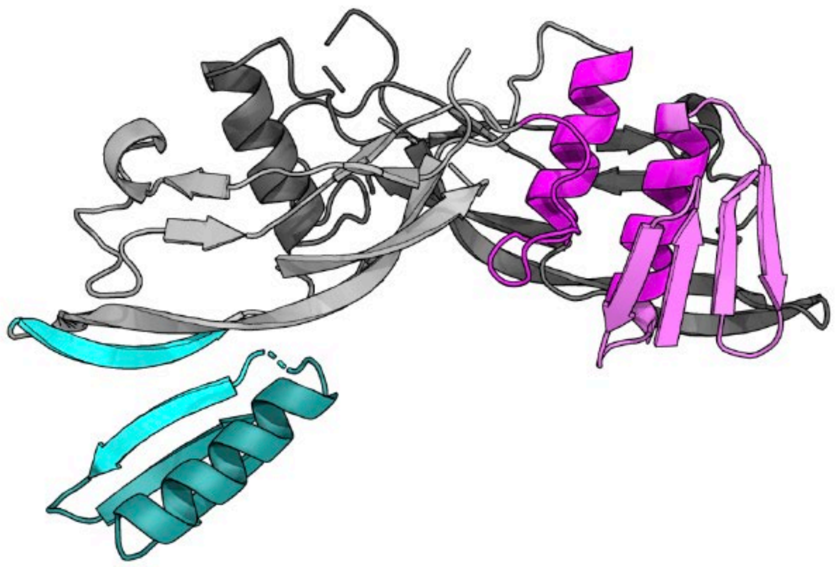


Figure 16. Top two final designs seeded with motifs that are predicted to bind the wrist and the knuckle.

Chapter 3: Machine Learning to Predict PPI Affinity

Introduction

Along with site-specificity, affinity is an important characteristic for fine-tuning the proposed biologic delivery mechanism to augment bone healing. However, experimental selection of known affinity binders is challenging. The computational design of protein binders with particular affinity is similarly elusive given the unclear relationship between modeled biophysical features of protein-protein interactions (ie. Rosetta metrics) and binding affinity. Recent advances in machine learning models and increases in the availability of accurate binding affinity data open up new possibilities for leveraging computational data science for protein design. Overall, compilation of datasets from biological experiments has generated a reckoning within the field of bioinformatics and protein design alike in translational tasks like classifying, predicting, and even generating proteins with functionality for a wide array of applications, from biotechnology to drug delivery.

Applying machine learning to the prediction of protein-protein interaction affinity could drive massive efforts to design reliable protein binders for biopharmaceuticals, biomaterials, and smart biologic diagnostics. In this project, I aimed to develop an affinity predictor to use in conjunction with my computational design pipeline or as part of a genetic algorithm in order to obtain binders. Here, I employed the PDBbind dataset of affinity-labeled protein-protein interactions from the Protein Data Bank to train a number of machine learning models to classify protein affinity into one of several quantitative categories with adequate accuracy. I propose that this model be used to iteratively guide designed proteins to obtain a desired affinity within a genetic algorithm to optimize protein binder selectivity, a step forward in the challenging field of negative protein design. For future improvement of the model, I suggest several ways in which the learning can be improved, including cross-validation, deep architecture, and larger and more diverse data input. Furthermore, a highly accurate affinity prediction model can be utilized in protein engineering efforts beyond the design of a biologic delivery mechanism for bone regeneration.

Methods

All code for dataset preparation, ML and DL models, and performance evaluation is in Appendix 1 under machine_learning.py.

Dataset

The PPI dataset was manually inspected by opening PDBs in PyMOL to find suitable protein interactions between two chains only. Metals and hetero-atoms were also excluded. The dataset of PDBs had 613 samples after this cleaning process. All PDBs were minimized with Rosetta FastRelax. The labels for this dataset correspond to four affinity bins split at experimental kd measurements included in the PDBbind dataset. If pkd, or the log of kd, is greater than or equal to 9, corresponding to a kd less than 1.0 nM, the sample is labeled class 1. If pkd is greater than or equal to 7 and less than 9, corresponding to a kd greater than 1.0 nM and less than 100 nM, the sample is labeled class 2. If pkd is greater than or equal to 5 and less than 7, corresponding to a kd greater than 100 nM and less than 10 μM, the sample is labeled class 3. Finally, if the pkd is less than 5, corresponding to a kd greater than 10 μM, the sample is labeled class 4.

Features

Interface analysis was then performed in Rosetta using the InterfaceAnalyzer mover on the dataset of relaxed protein-protein complexes to extract 76 features (Appendix 1; metrics_wet.xml).

Features:

'score', 'fa_atr', 'fa_rep', 'fa_sol', 'fa_intra_atr_xover4', 'fa_intra_rep_xover4', 'fa_intra_sol_xover4', 'lk_ball', 'lk_ball_iso', 'lk_ball_bridge', 'lk_ball_bridge_uncpl', 'fa_elec', 'fa_intra_elec', 'pro_close', 'hbond_sr_bb', 'hbond_lr_bb', 'hbond_bb_sc', 'hbond_sc', 'dslf_fa13', 'omega', 'fa_dun_dev', 'fa_dun_rot', 'fa_dun_semi', 'p_aa_pp', 'hxl_tors', 'ref', 'rama_prepro', 'gen_bonded', 'b_sap_score', 'bb_sap_score', 'buns_all', 'buns_bb', 'buns_sc', 'cms', 'complex_normalized', 'contact', 'contact_norm', 'dG_cross', 'dG_cross/dSASAx100', 'dG_separated', 'dG_separated/dSASAx100', 'dSASA_hphobic', 'dSASA_int', 'dSASA_polar', 'ddg_norep', 'ddg_rep', 'delta_unsatHbonds', 'filter_hyd_sasa', 'filter_pol_sasa', 'filter_sasa', 'h_sasa', 'hbond_E_fraction', 'hbonds_int', 'int_holes', 'int_hydcontact', 'mismatch_probability', 'nres_all', 'nres_int', 'p_sasa', 'packstat', 'per_residue_energy_int', 'pstat', 'sasa', 'sc', 'sc_value', 'side1_normalized', 'side1_score', 'side2_normalized', 'side2_score', 'ss_sc', 't_sap_score', 't_sasa', 'tb_sap_score', 'vbuns_all', 'vbuns_bb', ‘vbuns_sc'

For one model entitled ‘Random forest classifier with 8 features,’ feature engineering was performed by partitioning redundant features into 8 groups by general biophysical property (ie. solvent accessibility) and randomly choosing one feature from each partition on which to train the random forest classifier.

Partitions:

score_terms = ['total_score', 'per_residue_energy_int', 'side1_score', 'side2_score']
other_terms = ['fa_atr', 'fa_dun_dev', 'fa_dun_rot', 'fa_dun_semi', 'fa_elec', 'fa_intra_atr_xover4',
'fa_intra_elec', 'fa_intra_rep_xover4', 'fa_intra_sol_xover4', 'fa_rep', 'fa_sol', 'hxl_tors', 'lk_ball', 'lk_ball_bridge', 'lk_ball_bridge_uncpl', 'lk_ball_iso', 'omega', 'p_aa_pp', 'pro_close']
burial_terms = ['buns_all', 'buns_bb', 'buns_sc','vbuns_all', 'vbuns_bb', 'vbuns_sc','delta_unsatHbonds', 'hbond_E_fraction','hbonds_int','hbond_sr_bb', 'hbond_lr_bb', 'hbond_bb_sc', 'hbond_sc',]
packing_terms = ['cms']

ddg_terms = ['dG_cross', 'dG_separated', 'ddg_norep', ‘ddg_rep']
sasa_terms = ['dSASA_hphobic', 'dSASA_int', 'dSASA_polar', 'filter_hyd_sasa', 'filter_pol_sasa', 'filter_sasa', 'h_sasa','p_sasa','t_sasa', 'sasa']
sc_terms = ['ss_sc', 'mismatch_probability']
sap_terms = ['b_sap_score', 'bb_sap_score', 't_sap_score', 'tb_sap_score'] All other models used all 76 features.

ML Model

Train-test split, random forest regression, and performance metrics were built using the SKLearn python package. The model was written and tested in the Jupyter Notebooks environment. For random forests, 1000 trees were used and the models were built with default depth and ‘balanced’ class weight. A train/test split of 10% was used. For the gradient boosting classifier, 100 trees were used with a learning rate of 0.1 and a maximum depth of 4. For linear SVM, the linear kernel was used with a C value of 100. For rbf SVM, the rbf kernel was used with a C value of 0.1.

DL Model

Deep architectures included fully connected deep neural nets (DNNs) with five layers. The python Keras package was used to build and compile the models. In the DNN model, five dense layers constitute the model architecture, where the hidden layers used the relu activation function and the output layer used the sigmoid activation function. To train the DNN, 10 epochs were used with the Adam optimizer and the binary cross-entropy loss function. In the sequential 5-layer DNN, the sequential keras model is specified in the architecture and again, hidden layers use relu and the output layer uses the sigmoid activation function. To train the sequential DNN, 150 epochs were used with batch size of 10. The optimizer was Adam and categorical cross-entropy was used as the loss function. In both DNN models, the metric used to perform back-propagation was accuracy.

Performance metrics

Using SKLearn, accuracy, precision, recall, and f1 score were calculated on the test set. Confusion matrices, which are colored based on true positives, are provided to visualize performance of each model.

Results

Machine learning models are susceptible to bias depending on the dataset that is used for training. To investigate if my models may be biased by the representation of certain Kd class labels more than others, I analyzed the distribution of both Kd and class label in the cleaned dataset and found that both distributions were roughly normal (fig. 17). To re-weight the class labels, I used balanced class weighting in the random forest classifier with 76 features. Insufficient model robustness for predicting binding affinities of class label 1 and 4 may be the result of class weighting bias, demonstrated in the absence of true positives in the upper lefthand and lower righthand corners of the confusion matrices for machine learning models (fig. 20-24). Enlarging the dataset to offer a more uniform distribution of class labels may decrease this bias.

When trained on over 600 protein-protein interactions with over 70 biophysical and heuristic features, a random forest classifier model was able to achieve 56% accuracy, which is greater than the accuracy of the random forest classifier using 8 randomly selected features from 8 biophysical partitions. (fig. 25). Both classifiers, however, achieve greater accuracy than other models, including deep learning models.

Automated hyper-parameter tuning of the random forest classifier with 76 features may improve the accuracy of the model significantly. Currently, machine learning to predict binding affinity is far from robust or accurate, given that a random assignment of predicted labels to test samples would theoretically yield an accuracy of 50%. Relative feature importances are indicated in figure 19. Exploiting highly important features by engineering new features calculated from these may improve machine learning models.

An improved predictor may be further applied in order to guide the design of novel, selective binders. The machine learning model would be nested within a genetic algorithm (fig. 18). This genetic algorithm would diversify binder sequences while assessing the fitness of designs at each round by their higher affinity for BMP-2 than BMP-7, a close structural relative to BMP-2 and therefore an endogenous protein with the ability to generate off-target effects in the event of non-selective binding. The result of the genetic algorithm would therefore be sequences optimized for BMP-2 binding and unlikely to bind to BMP-7, ensuring the selectivity of binders. This is a process called multi-state design; on one hand, positive design increases the affinity toward one target, here, BMP-2 while negative design decreases affinity toward off-target proteins, like BMP-7.

Conclusion

I developed a machine learning model to predict the affinity of a protein-protein interaction based on biophysical and heuristic features which can be calculated by Rosetta. Improvement of the model, however, could be achieved using a larger dataset, hyper-parameter tuning, cross-validation (train/test/validation splitting instead of simply train/test splitting), and the addition of manually calculated, engineered features based on the metrics available through Rosetta. In fact, InterfaceAnalyzer is amenable to the engineering of new features using Calculator Metrics, or new metrics that are output via the calculation of one of more existing metrics within the script.

Overall, computational models enable efficient and cost-effective multi-state design pipelines. Experimental techniques such as magnetic bead sorting are also capable of multi-state selection, but these processes can be more labor intensive and can present the risk of decreasing, rather than increasing, the diversity of binders one can characterize downstream, thereby limiting access to different spaces in the sequence and structure landscape. Therefore, exploring machine learning methods in order to predict binding affinity is a worthwhile investigation, as it may contribute greatly to the robustness of functional protein binder design.

Figures

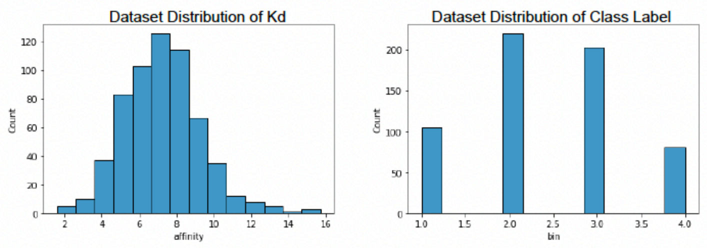


Figure 17. Distributions of Kd and class label in the cleaned PDBbind dataset used to train and test ML and DL models.

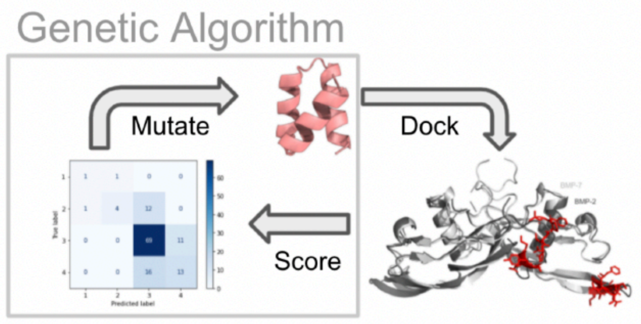


Figure 18. Schematic representation of ML model nested within a genetic algorithm for selective protein binder design. A binder is docked to two structurally similar proteins (ie. BMP-2 and BMP-7) producing a 'fitness score' for each. This fitness score is used to determine the parent pool of proteins with selective preference for BMP-2 over BMP-7. The ML model then predicts binding affinity to inform random mutation frequency in offspring proteins. These offspring are used as the inputs to the docking and continue the cycle.

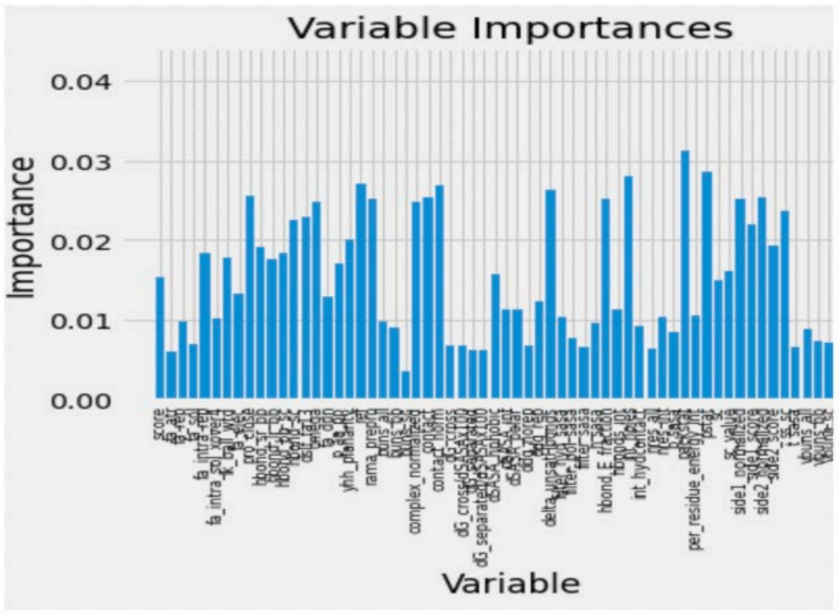


Figure 19. Feature importances calculated after model training for Random Forest Classifier with 76 Features using SKLearn.

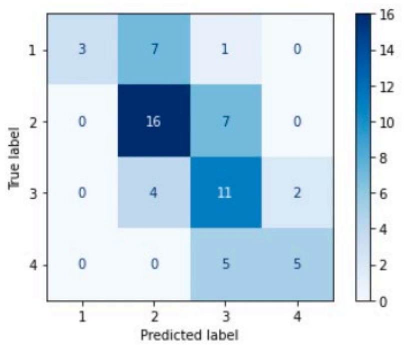


Figure 20. Confusion matrix for random forest classifier with 76 features.

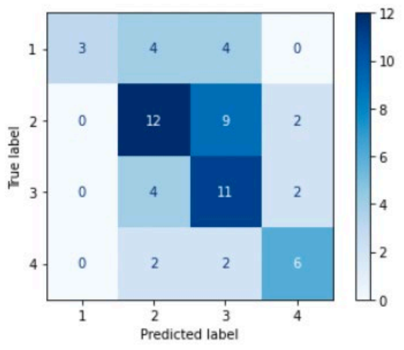


Figure 21. Confusion matrix for random forest classifier with 8 features.

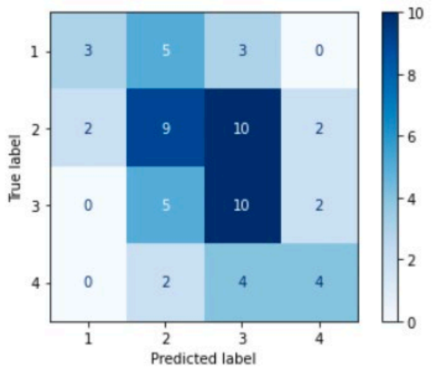


Figure 22. Confusion matrix for gradient boosting classifier.

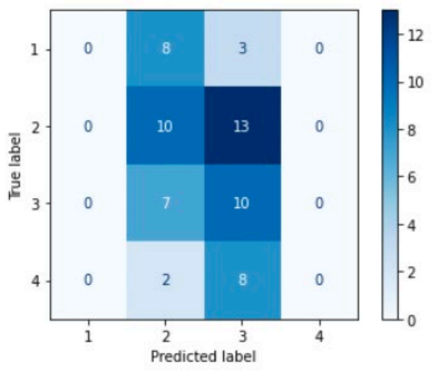


Figure 23. Confusion matrix for linear SVM.

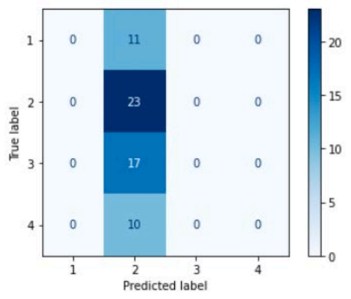


Figure 24. Confusion matrix for rbf SVM.

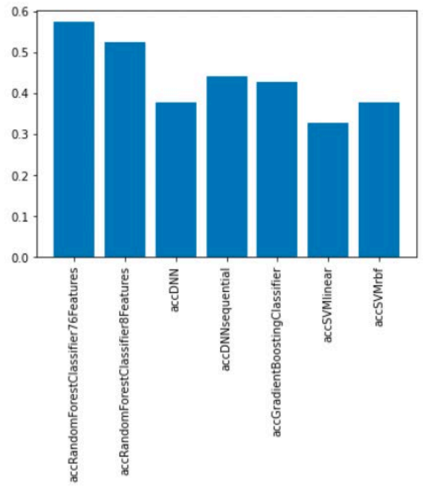


Figure 25. Accuracies of seven machine learning models trained on protein affinity data.

Discussion

Implications, Limitations, and Future Directions

Affinity interactions to fine-tune BMP-2 release into a critically sized bone defect can leverage the power of intentional or rational engineering for regenerative outcomes, where other methods like standard collagen sponges may fail to direct the signaling outcome of a potent biologic. In this project, I modeled and designed BMP-2 protein binders to control BMP-2 release from a hydrogel scaffold using computational pipelines. By incorporating force field approximations from Rosetta to modulate affibody affinity and rationally design site-specific binding proteins, I introduced structural insight into the design of affinity interactions.

The designs are therefore limited by the accuracy of structure prediction tools, crystal structures, and Rosetta in detailing protein and protein complex structure. The machine learning models I used to predict protein affinity are also limited due to the relatively small dataset (<700) of Rosetta-generated biophysical metrics used to train the models, which can be subject to the same inaccuracy as PPI models and designs and may be insufficient for learning.

In the future, I plan to explore other computational tools for protein modeling including molecular dynamics simulations and to increase the size of the current protein affinity dataset. I will further characterize purified BMP-2 protein binders, affibody mutants and novel designs alike, from E. Coli with size-exclusion chromatography, circular dichroism, X-ray crystallography, bioactivity assays and toxicity screening.

Binders that have been characterized as expressible, stable, and functional will be incorporated into BMP-2 delivery hydrogels and assessed in bone regeneration studies using a rat femoral fracture model. Fine-tuned BMP-2 delivery into large fractures may be augmented by both rationally designed BMP-2 protein binders and other engineered biomaterials. Fracture healing involves a suite of signaling proteins beside BMP-2 which are expressed and active during particular stages of bone regeneration, providing even more target proteins to potentially consider in the structure-guided design of affinity interactions. Therapeutic efficacy and regeneration may ultimately be achieved using a complex combination of bioengineered biologic release mechanisms and biomechanical scaffolds and materials.

Conclusion

I have validated the modeling and design of binder interactions to BMP-2 using experimental characterization in the form of yeast surface display and flow cytometry or FACS. In addition, I developed a machine learning model to predict protein affinity.

Within this project, I generated pipelines to model PPIs and ligand point mutants as well as rationally design specific protein binders. I intend to help train other members of the Hosseinzadeh lab and members of the Hettiaratchi lab and beyond to use these pipelines for their own applications in protein engineering, helping to grow the body of computationalists at the University of Oregon and even the Eugene-Springfield community, one workshop at a time.

In addition, I have been offered rewarding opportunities to present my research at the UO Undergraduate Research Symposium, RosettaCon, UO IMB Retreat, and the Oregon Bioengineering Symposium. Furthermore, I have had the opportunity to design a summer project for and mentor a high school student from the Camas High School STEM program as well as help train another undergraduate research assistant. My research journey has inspired me to co-found an undergraduate journal club and become an ASURE peer mentor. Undergraduate research has enabled me to program in multiple coding languages, conduct and troubleshoot experiments in the wet lab, communicate scientific research at varying levels of formality and to different kinds of audiences, and ultimately develop as a young scientist.

After graduation, I will matriculate into the Bioengineering PhD program at the Knight Campus where I will continue to grow and learn as a scientific researcher, communicator, writer, and mentor. I would not be in the position to write this thesis or

even pursue undergraduate research if not for all the wonderful people who have supported me. Thank you to my parents, my sisters, my friends, my lab-mates, my collaborators, my professors, and my advisors. I would like to especially thank my mentor, PI, and advisor, Parisa Hosseinzadeh, for all the time she has given me, all the support, encouragement, and feedback she has wholeheartedly offered me, and all the inspiration, as a person and as a scientist, she continues to provide me.

Appendix 1

XML and Python scripts

interface_analyzer.xml

```
<ROSETTASCRIPTS>

<SCOREFXNS>
<!--Defining the score function as the default-->
<ScoreFunction name="score" weights="ref2015.wts"/>
</SCOREFXNS>

<RESIDUE_SELECTORS>
<!--Defining the protein and target-->
<Chain name="bmp" chains="1"/>
<Chain name="binder" chains="2"/>
<!--Defining the protein, target interface-->
<Neighborhood name="bmp_binder_interface" selector="binder" distance="10."/>
</RESIDUE_SELECTORS>

<TASKOPERATIONS>
</TASKOPERATIONS>

<SIMPLE_METRICS>
<!--Sasa measurements,including polar and hydrophobic,for the interface-->
<SasaMetric name="tot_sasa"
residue_selector="bmp_binder_interface" sasa_metric_mode="all_sasa"/>
<SasaMetric name="pol_sasa" residue_selector="bmp_binder_interface" sasa_metric_mode="polar_sasa"/>
<SasaMetric name="hyd_sasa" residue_selector="bmp_binder_interface" sasa_metric_mode="hydrophobic_sasa"/>
</SIMPLE_METRICS>

<FILTERS>
<!--interface complementarity-->
<ShapeComplementarity name="shape_complement" min_sc="0.5" residue_selector1="binder"
                        residue_selector2="bmp" confidence="0"/>
<!--ddg with and without repack-->
<Ddg name="ddg_repack" chain_num="2" threshold="-1" repeats="5" repack="1" confidence="0" scorefxn="score"/>
<Ddg name="ddg_norepack" chain_num="2" threshold="-1" repack="0" confidence="0" scorefxn="score"/>
<!--filters cooresponding to our previous sasa metrics-->
<SimpleMetricFilter name="filter_sasa" metric="tot_sasa"
cutoff="100" comparison_type="gt"
confidence="0"/>
<SimpleMetricFilter name="filter_pol_sasa" metric="pol_sasa" cutoff="100" comparison_type="gt" confidence="0"/>
<SimpleMetricFilter name="filter_hyd_sasa" metric="hyd_sasa" cutoff="100" comparison_type="gt"
confidence="0"/>
<!--filtering buried unsatisfied polar atoms-->
<BuriedUnsatHbonds name="buried_unsat"
residue_selector="binder" report_all_heavy_atom_unsats="true" scorefxn="true" cutoff="4" ignore_surface_res="false"
print_out_info_to_pdb="true" use_ddG_style="true"
dalphaball_sasa="1" probe_radius="1.1" confidence="0" />
<BuriedUnsatHbonds name="very_buried_unsat"
residue_selector="binder" report_all_heavy_atom_unsats="true" scorefxn="score" ignore_surface_res="false"
print_out_info_to_pdb="true" atomic_depth_selection="5.5" burial_cutoff="1000" use_ddG_style="true"
burial_cutoff_apo="0.2" dalphaball_sasa="true" probe_radius="1.1" confidence="0" />

</FILTERS>

<MOVERS>
<!--Mover for the interface interaction-->
<InterfaceAnalyzerMover name="inter_move" scorefxn="score"
packstat="true" interface_sc="true" ligandchain="B"/>
```

```
<RunSimpleMetrics name="metric1" metrics="tot_sasa" prefix="m1_"/>
<RunSimpleMetrics name="metric2" metrics="pol_sasa" prefix="m2_"/>
<RunSimpleMetrics name="metric3" metrics="hyd_sasa" prefix="m3_"/>
</MOVERS>

<PROTOCOLS>
<!--Order of actions-->
<Add mover="inter_move"/>
<Add mover="metric1"/>
<Add mover="metric2"/>
<Add mover="metric3"/>
<Add filter="shape_complement"/>
<Add filter="ddg_repack"/>
<Add filter="ddg_norepack"/> Add filter="filter_sasa"/>
Add filter="filter_pol_sasa"/> Add filter="filter_hyd_sasa"/>
<Add filter="buried_unsat"/>
<Add filter="very_buried_unsat"/>
</PROTOCOLS>

<OUTPUT />
</ROSETTASCRIPTS>
```

greedy_opt_analysis.py

```
#!/usr/bin/env python # coding: utf-8
# # Greedy Opt Analysis

import pandas as pd import seaborn as sns import numpy as np

AA_list = [ "ARG",
"LYS",
"HIS",
"ASP",
"GLU",
"ASN",
"GLN",
"SER",
"THR",
"CYS",
"TRP",
"TYR",
"PHE",
"MET",
"ILE",
"LEU",
"VAL",
"ALA",
"PRO", "GLY"
]
number_native = [] sc_native=[] ddg_native=[]

num_lines = sum(1 for line in open('GreedyOptTable_g3c1_0577_0001_0001.tab')) print(num_lines)
sc_array=np.zeros((num_lines, len(AA_list))) ddg_array=np.zeros((num_lines, len(AA_list)))

# Filling in the dicts
with open("GreedyOptTable_g3c1_0577_0001_0001.tab") as f: #opening XXX.tab for line in f: # reading line by line
space_sep = line.rstrip().replace(
    '\t', ' ').split(" ") # splitting the line by space resi=space_sep[0] number_native.append(resi)
resi_point = len(number_native) - 1
for i in range(4,len(space_sep)): # 0-3 are resi, (resi, one-letter, and ). val_lines=space_sep[i].split(":") #getting 3-letter,
    ddg, -1*sc
if "*" in val_lines[0]:
sc_native.append(-1*float(val_lines[2])) ddg_native.append(float(val_lines[1])) AA3 = val_lines[0].replace("*","")
else:
    AA3 = val_lines[0] sc_array[resi_point][
              AA_list.index(AA3)] = -1*float(val_lines[2]) ddg_array[resi_point][
                    AA_list.index(AA3)]=float(val_lines[1]) sc_array[resi_point][:] -= sc_native[resi_point]
ddg_array[resi_point][:] -= ddg_native[resi_point]


sc_df = pd.DataFrame(sc_array,
columns=AA_list, index=number_native)
ddg_df = pd.DataFrame(ddg_array,
columns=AA_list, index=number_native)


sns.heatmap(sc_df) sns.heatmap(ddg_df, vmax=20) sns.heatmap(ddg_df, vmax=100)
sns.histplot(data=ddg_df.iloc[16,:], bins=10, binrange=(0,20)) sns.histplot(data=ddg_df.iloc[16,:], bins=10,
binrange=(20,100))
```

designsurface.xml

```
<ROSETTASCRIPTS>


<SCOREFXNS>
<ScoreFunction name="score" weights="ref2015.wts"/>
</SCOREFXNS>


<RESIDUE_SELECTORS>
<!--Selects the repeat protein scaffold-->
<Chain name="scaffold" chains="1"/>
<!--Selects anything within 8 of bmp as well as bmp-->
<Layer name="surface" select_surface="true" select_boundary="true" select_core="false" ball_radius="2"/>
<!--Selects the nonsurface for the taskop--> Not name="nonsurface" selector="surface"/>
        <ResidueName name="hyd" residue_names="LEU,ALA,VAL,ILE,GLY,PHE,TYR,TRP"/>
ResiduePropertySelector name="hyd" properties="HYDROPHOBIC"/> Not name="nonhyd" selector="hyd"/>
<And name="hyd_surface" selectors="surface,hyd"/>
<Not name="non_hs" selector="hyd_surface"/>
</RESIDUE_SELECTORS>


<TASKOPERATIONS>
<!--Prevent the core from designing-->
<OperateOnResidueSubset name="nodes_nonhs"
selector="non_hs">
<RestrictToRepackingRLT/>
</OperateOnResidueSubset>
</TASKOPERATIONS>


<FILTERS>
<ExposedHydrophobics name="exp_hyd" threshold="500.0" confidence="0"/>
</FILTERS>
<MOVE_MAP_FACTORIES>
<MoveMapFactory name="relax_mm" bb="false" chi="false" jumps="false">
<Chi residue_selector="non_hs"/>
<Backbone residue_selector="non_hs"/>
</MoveMapFactory>
</MOVE_MAP_FACTORIES>


<MOVERS>
        <FastDesign name="fast_design" scorefxn="score" task_operations="nodes_nonhs"
movemap_factory="relax_mm"/>
FavorNativeResidue name="nonnative_penalty" bonus="2"/>
</MOVERS>
<PROTOCOLS>
<Add mover="fast_design"/>
Add mover="nonnative_penalty"/>
<Add filter="exp_hyd"/>
```

```
</PROTOCOLS>

<OUTPUT />

</ROSETTASCRIPTS>
```

motif_graft.xml

```
<ROSETTASCRIPTS>

<SCOREFXNS>
<ScoreFunction name="score" weights="ref2015.wts"/>
</SCOREFXNS>

<!--This section defines selections from the pose-->
<RESIDUE_SELECTORS>
<!--This section defines the motif and the context-->
<Chain name="bmp" chains="1"/>
<Chain name="rgm_motif" chains="2"/>
</RESIDUE_SELECTORS>

<TASKOPERATIONS>
</TASKOPERATIONS>
<!--This section runs analysis and filters based on user-defined thresholds-->

<FILTERS>
<Sasa name="sasa" confidence="0"/>
<ShapeComplementarity name="sc" confidence="0"/>
<Ddg name="ddg" repack="0" jump="1" chain_num="2" confidence="0"/>
</FILTERS>
<!--This section contains blueprints for processes on the pose-->

<MOVERS>
<!--Grafts a motif structure into context structure-->
  <!--Removed hotspot residue SER8 from hotspots, inc mfrsd from -1:1, inc NCprt from 1.0, inc cfsd to 2:2 to increase
graft success (11/30/21)-->
<MotifGraft name="motif_grafting" context_structure="/home/kfear/parisahlab/project_bmp_binder/new_motifs/
bmp_for_grafting.pdb" motif_structure="/home/kfear/parisahlab/project_bmp_binder/new_motifs/
big_rgm_clean_relax.pdb" RMSD_tolerance="1.5"
  max_fragment_replacement_size_delta="-2:2" hotspots="4:6:7:11" NC_points_RMSD_tolerance="1.5"
clash_score_cutoff="5" clash_test_residue="ALA" full_motif_bb_alignment="1" revert_graft_to_native_sequence="1"
allow_repeat_same_graft_output="1" combinatory_fragment_size_delta="2:1"/> MultiplePoseMover
name="interface_design">
     xi:include href="/home/kfear/parisahlab/project_bmp_binder/scripts/ designinterface.xml"/>
/MultiplePoseMover>
  <!--Add backrub from backrub_interface_analysis.xml to increase scaffold diversity after grafting (1/5/22)-->
<BackrubDD name="backrub" partner1="1" partner2="1" interface_distance_cutoff="8.0" moves="10000"
  sc_move_probability="0.4" scorefxn="score"
small_move_probability="0.25" bbg_move_probability="0.25" temperature="0.6">
<span begin="1" end="55"/>
</BackrubDD>
</MOVERS>

<!--This section lays out the steps to run-->
<PROTOCOLS>
<Add mover="motif_grafting"/>
<Add mover="backrub"/>
Add mover="interface_design"/>
</PROTOCOLS>

<OUTPUT />

</ROSETTASCRIPTS>
```

backrub_grafts.xml

```
<ROSETTASCRIPTS>


<SCOREFXNS>
<!--Defining the score function as the default-->
<ScoreFunction name="score" weights="ref2015.wts"/>
</SCOREFXNS>


<RESIDUE_SELECTORS>
<!--Defining the protein and target-->
<Chain name="bmp" chains="1"/>
<Chain name="binder" chains="2"/>
<!--Defining the protein, target interface-->
<Neighborhood name="bmp_binder_interface" selector="binder" distance="5."/>
</RESIDUE_SELECTORS>


<TASKOPERATIONS>
</TASKOPERATIONS>


<SIMPLE_METRICS>
<!--Sasa measurements,including polar and hydrophobic,for the interface-->
<SasaMetric name="tot_sasa"
residue_selector="bmp_binder_interface" sasa_metric_mode="all_sasa"/>
<SasaMetric name="pol_sasa" residue_selector="bmp_binder_interface" sasa_metric_mode="polar_sasa"/>
<SasaMetric name="hyd_sasa" residue_selector="bmp_binder_interface" sasa_metric_mode="hydrophobic_sasa"/>
</SIMPLE_METRICS>


<FILTERS>
<!--interface complementarity-->
<ShapeComplementarity name="shape_complement" min_sc="0.5" residue_selector1="binder"
                              residue_selector2="bmp"
confidence="0"/>
<!--ddg with and without repack-->
<Ddg name="ddg_repack" chain_num="2" threshold="-1" repeats="5" repack="1" confidence="0" scorefxn="score"/>
<Ddg name="ddg_norepack" chain_num="2" threshold="-1" repack="0" confidence="0" scorefxn="score"/>
<!--filters cooresponding to our previous sasa metrics-->
<SimpleMetricFilter name="filter_sasa" metric="tot_sasa"
cutoff="100" comparison_type="gt" confidence="0"/>
<SimpleMetricFilter name="filter_pol_sasa" metric="pol_sasa" cutoff="100" comparison_type="gt" confidence="0"/>
<SimpleMetricFilter name="filter_hyd_sasa" metric="hyd_sasa" cutoff="100" comparison_type="gt"
confidence="0"/>
<!--filtering buried unsatisfied polar atoms-->
<BuriedUnsatHbonds name="buried_unsat"
residue_selector="binder" report_all_heavy_atom_unsats="true" scorefxn="true" cutoff="4" ignore_surface_res="false"
print_out_info_to_pdb="true" use_ddG_style="true"
dalphaball_sasa="1" probe_radius="1.1" confidence="0" />
<BuriedUnsatHbonds name="very_buried_unsat"
residue_selector="binder" report_all_heavy_atom_unsats="true" scorefxn="score" ignore_surface_res="false"
print_out_info_to_pdb="true" atomic_depth_selection="5.5" burial_cutoff="1000" use_ddG_style="true"
burial_cutoff_apo="0.2" dalphaball_sasa="true" probe_radius="1.1" confidence="0" />
</FILTERS>


<MOVERS>
<!--Mover for the interface interaction-->
<InterfaceAnalyzerMover name="inter_move" scorefxn="score"
packstat="true" interface_sc="true" ligandchain="B"/>
<RunSimpleMetrics name="metric1" metrics="tot_sasa" prefix="m1_"/>
<RunSimpleMetrics name="metric2" metrics="pol_sasa" prefix="m2_"/>
<RunSimpleMetrics name="metric3" metrics="hyd_sasa" prefix="m3_"/>
        <BackrubDD name="backrub" partner1="1" partner2="1" interface_distance_cutoff="8.0" moves="10000"
sc_move_probability="0.4" scorefxn="score" small_move_probability="0.25" bbg_move_probability="0.25"
temperature="0.6">
       <span begin="%%first%%" end="%%last%%" /> <!--span takes variables from the batch file assigned upon reading
the pdb. Required for diversity.-->
</BackrubDD>
</MOVERS>
```

```
<PROTOCOLS>
<!--Order of actions-->
<Add mover="inter_move"/>
<Add mover="backrub"/>
<Add mover="metric1"/>
<Add mover="metric2"/>
<Add mover="metric3"/>
<Add filter="shape_complement"/>
<Add filter="ddg_repack"/>
<Add filter="ddg_norepack"/>
<Add filter="filter_sasa"/>
<Add filter="filter_pol_sasa"/>
<Add filter="filter_hyd_sasa"/>
<Add filter="buried_unsat"/>
<Add filter="very_buried_unsat"/>
</PROTOCOLS>

<OUTPUT />


</ROSETTASCRIPTS>
```

optimize_sequence.xml

```
<ROSETTASCRIPTS>
<!--Author: Karly Fear, Hosseinzadeh Lab, University of Oregon.-->
<!--Most recently updated script with more comments on 4/18/22-->
<!--Updated script for October/November 2021 interface design. Previous script: designinterface.xml-->


<SCOREFXNS>
<ScoreFunction name="score" weights="ref2015.wts"/>
<ScoreFunction name="beta" weights="beta_genpot.wts"/>
<ScoreFunction name="ICO" weights="beta_nov16.wts"/>
</SCOREFXNS>


<RESIDUE_SELECTORS>
<!--Selects the target protein and the binding protein. (A+B) and (C+D)-->
<Chain name="target" chains="1"/>
<Chain name="binder" chains="2"/>
<!--Selects anything within 8 of bmp as well as bmp. (A+B+C)-->
<Neighborhood name="target_neighborhood" selector="target" distance="8.0"/>
<!--Selects only ligand interface residues. (C)-->
<And name="binder_interface" selectors="binder,target_neighborhood"/>
<!--Selects anything within 8 of the binder as well as the binder. (B+C+D)-->
<Neighborhood name="binder_neighborhood" selector="binder" distance="8.0"/>
<!--Selects the interface residues of both proteins. (B+C)-->
<And name="int" selectors="target_neighborhood,binder_neighborhood"/>
<!--Selects the residues of both proteins that are not at the interface-->
<Not name="not_int" selector="int"/>
<!--Selects the target interface. (B)-->
<And name="target_interface" selectors="target,binder_neighborhood"/>
<!--Selects hydrophobic residues-->
<ResiduePropertySelector name="hyd" properties="HYDROPHOBIC"/>
<!--Selects hydrophobic interface residues of the target protein-->
<And name="target_interface_hyd" selectors="hyd,target_interface"/>
<!--Selects all that is not the target interface. (A+C+D)-->
<Not name="target_noninterface" selector="target_interface"/>
<!--Defining the residues that are not at the interface. (A+B+D)-->
<Not name="binder_noninterface" selector="binder_interface"/>
<!--Selects the binder residues that are not at the interface. (D)-->
        <And name="binder_noninterface_without_target" selectors="binder,binder_noninterface"/>
<!--Selects the target residues that are not at the interface. (A)-->
        <And name="target_noninterface_without_binder" selectors="target,target_noninterface"/>
<!--binder hotspots-->
<ResiduePDBInfoHasLabel name="hotspots" property="HOTSPOT"/>
<!--core-->
<Layer name="core" select_core="true" ball_radius="2.5"/>
<True name="true_sel"/>
```

```
<!--glycines and prolines-->
<ResidueName name="GLY_PRO" residue_names="GLY,PRO"/>
</RESIDUE_SELECTORS>

<TASKOPERATIONS>
<!--Task = do not repack the target noninterface-->
<OperateOnResidueSubset name="hold_target_noninterface"
selector="target_noninterface_without_binder">
<PreventRepackingRLT/>
</OperateOnResidueSubset>
<!--Task = do not design the target interface-->
<OperateOnResidueSubset name="only_repack_target_interface" selector="target_interface">
<RestrictToRepackingRLT/>
</OperateOnResidueSubset>
<!--Task = do not design hotspots-->
<OperateOnResidueSubset name="only_repack_hotspots" selector="hotspots">
<RestrictToRepackingRLT/>
</OperateOnResidueSubset>
<!--Task = do not design binder noninterface-->
<OperateOnResidueSubset name="hold_binder_noninterface" selector="binder_noninterface_without_target">
<RestrictToRepackingRLT/>
</OperateOnResidueSubset>
<!--Task = do not design protein core-->
<OperateOnResidueSubset name="hold_core" selector="core">
<RestrictToRepackingRLT/>
</OperateOnResidueSubset>
<!--Task = do not design glycines and prolines-->
<OperateOnResidueSubset name="hold_gly_pro" selector="GLY_PRO">
<RestrictToRepackingRLT/>
</OperateOnResidueSubset>
</TASKOPERATIONS>


<TASKOPERATIONS>
<!--Add weight to the PPI in the score function. Should be at least 2.0-->
<ProteinProteinInterfaceUpweighter name="upweight" interface_weight="3.0"/>
<!--Helps with design-->
<ExtraRotamersGeneric name="extra_chi" ex1="1" ex2="1" extrachi_cutoff="0"/>
<RestrictToRepacking name="restrict"/>
<InitializeFromCommandline name="init"/>
    <RestrictToInterfaceVector name="intonly" chain1_num="1" chain2_num="2" CB_dist_cutoff="10.0"
      nearby_atom_cutoff="5.5" vector_angle_cutoff="75.0" vector_dist_cutoff="9.0" include_all_water="1"/>
</TASKOPERATIONS>


<SIMPLE_METRICS>
    <!--Sasa (solvent accessible surface area) measurements,including polar and hydrophobic,for the interface-->
    <SasaMetric name="tot_sasa"         residue_selector="int" sasa_metric_mode="all_sasa"/>
    <SasaMetric name="pol_sasa"         residue_selector="int" sasa_metric_mode="polar_sasa"/>
<SasaMetric name="hyd_sasa"             residue_selector="int" sasa_metric_mode="hydrophobic_sasa"/>
</SIMPLE_METRICS>


<FILTERS>
<!--geometric interface complementarity-->
    <ShapeComplementarity name="sc" min_sc="0.5" residue_selector1="target" residue_selector2="binder"
jump="1" confidence="0"/>
<!--ddg (delta-delta-G, free folding energy) with and without repack-->
<Ddg name="ddg_rep" chain_num="2" threshold="-1" jump="1" repeats="5" repack="1" confidence="0"
  scorefxn="score"/>
<Ddg name="ddg_norep" chain_num="2" threshold="-1" jump="1" repack="0" confidence="0" scorefxn="score"/>
<!--filters cooresponding to our previous sasa metrics-->
    <SimpleMetricFilter name="filter_sasa" metric="tot_sasa" cutoff="100" comparison_type="gt" confidence="0"/>
    <SimpleMetricFilter name="filter_pol_sasa" metric="pol_sasa" cutoff="100" comparison_type="gt" confidence="0"/>
    <SimpleMetricFilter name="filter_hyd_sasa" metric="hyd_sasa" cutoff="100" comparison_type="gt"
confidence="0"/>
<!--filtering for buried unsatisfied polar atoms-->
<BuriedUnsatHbonds name="buns_bb" residue_selector="int" report_bb_heavy_atom_unsats="true" scorefxn="true"
  cutoff="10" ignore_surface_res="false"
print_out_info_to_pdb="true" use_ddG_style="true" jump_number="1" dalphaball_sasa="1" probe_radius="1.1"
confidence="0"/>
```

```
<BuriedUnsatHbonds name="buns_sc" residue_selector="int" report_sc_heavy_atom_unsats="true" scorefxn="true"
  cutoff="10" ignore_surface_res="false"
print_out_info_to_pdb="true" use_ddG_style="true" jump_number="1" dalphaball_sasa="1" probe_radius="1.1"
confidence="0"/>
<BuriedUnsatHbonds name="buns_all" residue_selector="int" report_all_heavy_atom_unsats="true" scorefxn="true"
  cutoff="10" ignore_surface_res="false"
print_out_info_to_pdb="true" use_ddG_style="true" jump_number="1" dalphaball_sasa="1" probe_radius="1.1"
confidence="0"/>
<!--filtering for VERY buried unsatisfied polar atoms, harsher than buns-->
<BuriedUnsatHbonds name="vbuns_bb" residue_selector="int" report_bb_heavy_atom_unsats="true"
scorefxn="score" ignore_surface_res="false" print_out_info_to_pdb="true" atomic_depth_selection="5.5"
burial_cutoff="1000" use_ddG_style="true" jump_number="1" burial_cutoff_apo="0.2" dalphaball_sasa="true"
probe_radius="1.1" confidence="0" />
<BuriedUnsatHbonds name="vbuns_sc" residue_selector="int" report_sc_heavy_atom_unsats="true" scorefxn="score"
  ignore_surface_res="false" print_out_info_to_pdb="true" atomic_depth_selection="5.5" burial_cutoff="1000"
  use_ddG_style="true" jump_number="1" burial_cutoff_apo="0.2" dalphaball_sasa="true"
probe_radius="1.1" confidence="0" />
<BuriedUnsatHbonds name="vbuns_all" residue_selector="int" report_all_heavy_atom_unsats="true" scorefxn="score"
  ignore_surface_res="false" print_out_info_to_pdb="true" atomic_depth_selection="5.5" burial_cutoff="1000"
  use_ddG_style="true" jump_number="1" burial_cutoff_apo="0.2" dalphaball_sasa="true" probe_radius="1.1"
  confidence="0" />
<!--Packing statistics-->
<PackStat name="pstat" threshold="0.65" chain="2" confidence="0"/>
<!--Holes at the protein-protein interface-->
<InterfaceHoles name="int_holes" jump="1" confidence="0"/>
<!--Secondary structures at the interface-->
<SSShapeComplementarity name="ss_sc" verbose="1" loops="1" helices="1" min_sc="0.5" confidence="0"/>
<!--Holes or cavities of the binder-->
<Holes name="holes" threshold="1.0" residue_selector="binder"/>
<!--Interface contact-->
<InterfaceHydrophobicResidueContacts name="int_hydcontact" target_selector="target_interface_hyd"
  binder_selector="binder_interface"
scorefxn="score" apolar_res="ALA,CYS,CYD,PHE,ILE,LEU,MET,PRO,THR,VAL,TRP,TYR"
confidence="0"/>
<AtomicContactCount name="contact" partition="jump" jump="1" normalize_by_sasa="0" confidence="0"/>
<AtomicContactCount name="contact_norm" partition="jump" jump="1" normalize_by_sasa="1" confidence="0"/>
<Sasa name="sasa" confidence="0"/>
    <SSPrediction name="mismatch_probability" cmd="/projects/parisahlab/kfear/
project_bmp_binder/psipred/runpsipred_single"
use_probability="1" mismatch_probability="1" use_svm="0" confidence="0"/>
<!--Packing-->
<ContactMolecularSurface name="cms" distance_weight="0.5" target_selector="target" binder_selector="binder"
  confidence="0"/>
</FILTERS>


<SIMPLE_METRICS>
<!--Hydrophobicity metrics-->
<SapScoreMetric name="binder_sap" score_selector="target"/>
<SapScoreMetric name="target_sap" score_selector="binder"/>
<SapScoreMetric name="binder_blocked_sap" score_selector="target" sap_calculate_selector="binder"
  sasa_selector="true_sel"/>
<SapScoreMetric name="target_blocked_sap" score_selector="binder" sap_calculate_selector="target"
  sasa_selector="true_sel"/>
    <CalculatorMetric name="binder_delta_sap" equation="binder_sap_score - binder_blocked_sap">
<VAR name="binder_sap_score" metric="binder_sap"/>
<VAR name="binder_blocked_sap" metric="binder_blocked_sap"/>
</CalculatorMetric>
<CalculatorMetric name="target_delta_sap" equation="target_sap_score - target_blocked_sap">
<VAR name="target_sap_score" metric="target_sap"/>
<VAR name="target_blocked_sap" metric="target_blocked_sap"/>
</CalculatorMetric>
</SIMPLE_METRICS>


<MOVERS>
<!--Mover for relax/minimization before running metrics-->
<FastRelax name="relax" scorefxn="score" disable_design="1"
repeats="10" ramp_down_constraints="1" relaxscript="InterfaceRelax2019">
<MoveMap name="relax_mm" bb="1" chi="1" jump="1">
```

```
          <ResidueSelector selector="not_int" chi="0" bb="0" bondangle="0" bondlength="0"/>
<Jump number="1" setting="true"/>
</MoveMap>
</FastRelax>
<!--Mover for the interface analysis that helps with filtering-->
<InterfaceAnalyzerMover name="inter_move" scorefxn="score" packstat="true" interface_sc="true" jump="1"/>
<ddG name="ddg_move" scorefxn="ICO" chain_num="2" solvate="1" repack_bound="1" repack_unbound="1"
  solvate_rbmin="0" solvate_unbound="0" min_water_jump="1" task_operations="intonly,restrict,extra_chi"/>
<RunSimpleMetrics name="metric1" metrics="tot_sasa" prefix="t_"/>
<RunSimpleMetrics name="metric2" metrics="pol_sasa" prefix="p_"/>
<RunSimpleMetrics name="metric3" metrics="hyd_sasa" prefix="h_"/>
<RunSimpleMetrics name="bsap" metrics="binder_sap" prefix="b_"/>
<RunSimpleMetrics name="tsap" metrics="target_sap" prefix="t_"/>
      <RunSimpleMetrics name="bblock_sap" metrics="binder_blocked_sap" prefix="bb_"/>
<RunSimpleMetrics name="tblock_sap" metrics="target_blocked_sap" prefix="tb_"/
>
<RunSimpleMetrics name="b_delta_sap" metrics="binder_delta_sap" prefix="bd_"/
>
<RunSimpleMetrics name="t_delta_sap" metrics="target_delta_sap" prefix="td_"/>
</MOVERS>

<MOVERS>
<!--Mover for design/sequence optimization-->
<!--could add extra_chi to task operations (noted 4/18/22)-->
<FastDesign name="fast_design" scorefxn="score"
relaxscript="InterfaceDesign2019"
task_operations="hold_target_noninterface,hold_binder_noninterface,only_repack_targ
et_interface,only_repack_hotspots,upweight,hold_core,hold_gly_pro"
movemap_factory="relax_mm"/>
<!--Increased penalty for mutating the sequence-->
<FavorNativeResidue name="nonnative_penalty" bonus="1.5"/>
</MOVERS>

<PROTOCOLS>
<!--protocols for running sequence optimization-->
<!--1.relax/minimize structures 2.optimize sequence
3.filter designs -->
<Add mover="relax"/>
<Add mover="fast_design"/>
<Add mover="nonnative_penalty"/>
<Add mover="inter_move"/>
<Add filter="sc"/>
<Add filter="ddg_rep"/>
<Add filter="ddg_norep"/>
<Add filter="filter_sasa"/>
<Add filter="filter_pol_sasa"/>
<Add filter="filter_hyd_sasa"/>
<Add filter="buns_bb"/>
<Add filter="buns_sc"/>
<Add filter="buns_all"/>
<Add filter="vbuns_bb"/>
<Add filter="vbuns_sc"/>
<Add filter="vbuns_all"/>
<Add filter="pstat"/>
<Add filter="int_holes"/>
<Add filter="ss_sc"/>
<Add filter="int_hydcontact"/>
<Add filter="holes"/>
<Add filter="contact"/>
<Add filter="contact_norm"/>
<Add filter="sasa"/>
<Add filter="mismatch_probability"/>
<Add filter="cms"/>
<Add mover="bsap"/>
<Add mover="tsap"/>
<Add mover="bblock_sap"/>
<Add mover="tblock_sap"/>
<Add mover="b_delta_sap"/>
```

```
<Add mover="t_delta_sap"/>
<!--metrics added in 4/18/22 update-->
<Add metric="binder_sap"/>
<Add metric="target_sap"/>
<Add metric="binder_blocked_sap"/>
<Add metric="target_blocked_sap"/>
<Add metric="binder_delta_sap"/>
<Add metric="target_delta_sap"/>
</PROTOCOLS>

<OUTPUT />

</ROSETTASCRIPTS>
```

metrics_wet.xml

```
<ROSETTASCRIPTS>

<SCOREFXNS>
<!--Defining the score function as the default-->
<ScoreFunction name="score" weights="beta_genpot.wts"/>
<ScoreFunction name="ICO" weights="beta_nov16.wts"/>
</SCOREFXNS>

<RESIDUE_SELECTORS>
<!--Defining the protein and target-->
<Chain name="A" chains="1"/>
<Chain name="B" chains="2"/>
<!--Defining the protein, target interface-->
        <Neighborhood name="A_int" selector="A" distance="8.0"/>
<Neighborhood name="B_int" selector="B" distance="8.0"/>
<ResiduePropertySelector name="hyd" properties="HYDROPHOBIC"/>
<And name="int" selectors="A_int,B_int"/>
<Not name="not_int" selector="int"/>
<And name="int_A" selectors="A,int"/>
<And name="int_B" selectors="B,int"/>
<And name="hyd_int_A" selectors="hyd,int_A"/>
<True name="true_sel"/>
</RESIDUE_SELECTORS>

<TASKOPERATIONS>
<ExtraRotamersGeneric name="extra_chi" ex1="1" ex2="1" extrachi_cutoff="0"/>
<RestrictToRepacking name="restrict"/>
<InitializeFromCommandline name="init"/>
      <RestrictToInterfaceVector name="intonly" chain1_num="1" chain2_num="2" CB_dist_cutoff="10.0"
        nearby_atom_cutoff="5.5" vector_angle_cutoff="75.0" vector_dist_cutoff="9.0" include_all_water="1"/>
</TASKOPERATIONS>

<SIMPLE_METRICS>
<!--Sasa measurements,including polar and hydrophobic,for the interface-->
<SasaMetric name="tot_sasa" residue_selector="int" sasa_metric_mode="all_sasa"/>
        <SasaMetric name="pol_sasa" residue_selector="int"
sasa_metric_mode="polar_sasa"/>
        <SasaMetric name="hyd_sasa" residue_selector="int" sasa_metric_mode="hydrophobic_sasa"/>
</SIMPLE_METRICS>

<FILTERS>
<!--interface complementarity-->
<ShapeComplementarity name="sc" min_sc="0.5" residue_selector1="A" residue_selector2="B"
jump="1" confidence="0"/>
<!--ddg with and without repack-->
<Ddg name="ddg_rep" chain_num="2" threshold="-1" jump="1" repeats="5" repack="1" confidence="0"
  scorefxn="score"/>
        <Ddg name="ddg_norep" chain_num="2" threshold="-1" jump="1" repack="0" confidence="0" scorefxn="score"/>
<!--filters cooresponding to our previous sasa metrics-->
<SimpleMetricFilter name="filter_sasa" metric="tot_sasa" cutoff="100" comparison_type="gt" confidence="0"/>
```

```
        <SimpleMetricFilter name="filter_pol_sasa" metric="pol_sasa" cutoff="100" comparison_type="gt"
confidence="0"/>
        <SimpleMetricFilter name="filter_hyd_sasa" metric="hyd_sasa" cutoff="100" comparison_type="gt"
confidence="0"/>
<!--filtering buried unsatisfied polar atoms-->
<BuriedUnsatHbonds name="buns_bb"
residue_selector="int" report_bb_heavy_atom_unsats="true" scorefxn="true" cutoff="10" ignore_surface_res="false"
print_out_info_to_pdb="true" use_ddG_style="true"
jump_number="1"
dalphaball_sasa="1" probe_radius="1.1" confidence="0"/>
<BuriedUnsatHbonds name="buns_sc" residue_selector="int" report_sc_heavy_atom_unsats="true" scorefxn="true"
  cutoff="10" ignore_surface_res="false"
print_out_info_to_pdb="true" use_ddG_style="true" jump_number="1" dalphaball_sasa="1" probe_radius="1.1"
confidence="0"/>
<BuriedUnsatHbonds name="buns_all" residue_selector="int" report_all_heavy_atom_unsats="true" scorefxn="true"
  cutoff="10" ignore_surface_res="false"
print_out_info_to_pdb="true" use_ddG_style="true" jump_number="1" dalphaball_sasa="1" probe_radius="1.1"
confidence="0"/>
<BuriedUnsatHbonds name="vbuns_bb"
residue_selector="int" report_bb_heavy_atom_unsats="true" scorefxn="score" ignore_surface_res="false"
print_out_info_to_pdb="true" atomic_depth_selection="5.5" burial_cutoff="1000" use_ddG_style="true"
jump_number="1" burial_cutoff_apo="0.2" dalphaball_sasa="true" probe_radius="1.1" confidence="0" />
<BuriedUnsatHbonds name="vbuns_sc"
residue_selector="int" report_sc_heavy_atom_unsats="true" scorefxn="score" ignore_surface_res="false"
print_out_info_to_pdb="true" atomic_depth_selection="5.5" burial_cutoff="1000" use_ddG_style="true"
jump_number="1" burial_cutoff_apo="0.2" dalphaball_sasa="true" probe_radius="1.1" confidence="0" />
<BuriedUnsatHbonds name="vbuns_all" residue_selector="int" report_all_heavy_atom_unsats="true" scorefxn="score"
  ignore_surface_res="false" print_out_info_to_pdb="true" atomic_depth_selection="5.5" burial_cutoff="1000"
  use_ddG_style="true" jump_number="1" burial_cutoff_apo="0.2" dalphaball_sasa="true" probe_radius="1.1"
  confidence="0" />
<!--Packing statistics-->
<PackStat name="pstat" threshold="0.65" chain="1" confidence="0"/>
<!--Holes at the protein-protein interface-->
<InterfaceHoles name="int_holes" jump="1" confidence="0"/>
<!--Secondary structures at the interface-->
<SSShapeComplementarity name="ss_sc" verbose="1" loops="1" helices="1" min_sc="0.5" confidence="0"/>
<!--Interface contact-->
<InterfaceHydrophobicResidueContacts name="int_hydcontact" target_selector="hyd_int_A" binder_selector="int_B"
  scorefxn="score" apolar_res="ALA,CYS,CYD,PHE,ILE,LEU,MET,PRO,THR,VAL,TRP,TYR"
confidence="0"/>
<AtomicContactCount name="contact" partition="jump" jump="1" normalize_by_sasa="0" confidence="0"/>
<AtomicContactCount name="contact_norm" partition="jump" jump="1" normalize_by_sasa="1" confidence="0"/>
<Sasa name="sasa" confidence="0"/>
       <SSPrediction name="mismatch_probability" cmd="/projects/parisahlab/kfear/
project_bmp_binder/psipred/runpsipred_single"
                    use_probability="1" mismatch_probability="1" use_svm="0" confidence="0"/>
<!--Packing-->
<ContactMolecularSurface name="cms" distance_weight="0.5" target_selector="B" binder_selector="A"
                      confidence="0"/>
</FILTERS>


<SIMPLE_METRICS>
<SapScoreMetric name="binder_sap" score_selector="A"/>
<SapScoreMetric name="target_sap" score_selector="B"/>
<SapScoreMetric name="binder_blocked_sap" score_selector="A" sap_calculate_selector="A"
  sasa_selector="true_sel"/>
<SapScoreMetric name="target_blocked_sap" score_selector="B" sap_calculate_selector="B"
  sasa_selector="true_sel"/>
<CalculatorMetric name="binder_delta_sap" equation="binder_sap_score - binder_blocked_sap">
<VAR name="binder_sap_score" metric="binder_sap"/>
<VAR name="binder_blocked_sap" metric="binder_blocked_sap"/>
</CalculatorMetric>
      <CalculatorMetric name="target_delta_sap" equation="target_sap_score - target_blocked_sap">
<VAR name="target_sap_score" metric="target_sap"/>
<VAR name="target_blocked_sap" metric="target_blocked_sap"/>
</CalculatorMetric>
</SIMPLE_METRICS>
```

```
<MOVERS>
<!--Relax before running metrics-->
<FastRelax name="relax" scorefxn="score" disable_design="1"
repeats="10" ramp_down_constraints="1" relaxscript="InterfaceRelax2019">
<MoveMap name="relax_mm" bb="1" chi="1" jump="0">
         <ResidueSelector selector="not_int" chi="0" bb="0" bondangle="0" bondlength="0"/>
</MoveMap>
</FastRelax>
<!--Mover for the interface interaction-->
        <InterfaceAnalyzerMover name="inter_move" scorefxn="score" packstat="true" interface_sc="true" jump="1"/>
<ddG name="ddg_move" scorefxn="ICO" chain_num="1" solvate="1" repack_bound="1" repack_unbound="1"
  solvate_rbmin="0" solvate_unbound="0" min_water_jump="1" task_operations="intonly,restrict,extra_chi"/>
<RunSimpleMetrics name="metric1" metrics="tot_sasa" prefix="t_"/>
<RunSimpleMetrics name="metric2" metrics="pol_sasa" prefix="p_"/>
<RunSimpleMetrics name="metric3" metrics="hyd_sasa" prefix="h_"/>
<RunSimpleMetrics name="bsap" metrics="binder_sap" prefix="b_"/>
<RunSimpleMetrics name="tsap" metrics="target_sap" prefix="t_"/>
      <RunSimpleMetrics name="bblock_sap" metrics="binder_blocked_sap" prefix="bb_"/>
<RunSimpleMetrics name="tblock_sap" metrics="target_blocked_sap" prefix="tb_"/>

</MOVERS>

<PROTOCOLS>
<!--Order of actions-->
<Add mover="relax"/>
<Add mover="inter_move"/>
<Add mover="metric1"/>
<Add mover="metric2"/>
<Add mover="metric3"/>
<Add mover="bsap"/>
<Add mover="tsap"/>
<Add mover="bblock_sap"/>
<Add mover="tblock_sap"/>
<Add filter="filter_sasa"/>
<Add filter="filter_pol_sasa"/>
<Add filter="filter_hyd_sasa"/>
<Add filter="sc"/>
<Add filter="ddg_rep"/>
<Add filter="ddg_norep"/>
<Add filter="buns_bb"/>
<Add filter="buns_sc"/>
<Add filter="buns_all"/>
<Add filter="vbuns_bb"/>
<Add filter="vbuns_sc"/>
<Add filter="vbuns_all"/>
<Add filter="pstat"/>
<Add filter="int_holes"/>
<Add filter="ss_sc"/>
<Add filter="int_hydcontact"/>
<Add filter="contact"/>
<Add filter="contact_norm"/>
<Add filter="sasa"/>
<Add filter="mismatch_probability"/>
<Add filter="cms"/>
</PROTOCOLS>

<OUTPUT />

</ROSETTASCRIPTS>
```

machine_learning.py

```
#!/usr/bin/env python # coding: utf-8
import pandas as pd import seaborn as sns import numpy as np
import matplotlib.pyplot as plt from scipy.stats import norm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, precision_score, f1_score, recall_score from sklearn.model_selection
import train_test_split
```

```
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier import random

#read in the score file
rel = pd.read_csv("relaxed_2021_10_05.dat",sep="\s+") rel = rel.drop([491,500,524])

#read in the excel spreadsheet
csv = pd.read_csv('PDBbind.csv', header=1)[['PDB code','pKd pKi pIC50']]

#create name column from csv and convert to list name = csv['PDB code']+'_0001'
lname = name.tolist()

#make list from description column of rel ldescription = rel['description'].tolist()

#loop for the shared indeces shared = []
for i in range(len(lname)):
if lname[i] in ldescription: shared.append(i)

#loop to get the affinity values at the name indeces affinity = csv['pKd pKi pIC50']
lines_of_csv = [] for index in shared:
lines_of_csv.append(affinity[index])

#add affinity data from csv to relaxed dataframe rel['affinity'] = lines_of_csv

#add bins
aff = rel['affinity'] bins = []
for val in aff: if val >= 9:
bins.append(1)
elif (val >= 7) & (val < 9):
bins.append(2)
elif (val >= 5) & (val < 7): bins.append(3)
elif val < 5: bins.append(4)
rel['bin'] = bins

# copy the data df_sklearn = rel.copy()

#normalize using Min-Max scaler for col in rel.columns[1:-6]:
   df_sklearn[col] = MinMaxScaler().fit_transform(np.array(df_sklearn[col]).reshape(-1,1))

#separate labels (the values we want to predict) and features labels = np.array(df_sklearn['bin'])

features = df_sklearn[['score', 'fa_atr', 'fa_rep', 'fa_sol', 'fa_intra_atr_xover4', 'fa_intra_rep_xover4', 'fa_intra_sol_xover4',
       'lk_ball', 'lk_ball_iso', 'lk_ball_bridge', 'lk_ball_bridge_uncpl', 'fa_elec', 'fa_intra_elec', 'pro_close', 'hbond_sr_bb',
       'hbond_lr_bb', 'hbond_bb_sc', 'hbond_sc', 'dslf_fa13', 'omega', 'fa_dun_dev', 'fa_dun_rot', 'fa_dun_semi', 'p_aa_pp',
       'hxl_tors', 'ref', 'rama_prepro', 'gen_bonded',
'b_sap_score', 'bb_sap_score', 'buns_all', 'buns_bb', 'buns_sc', 'cms', 'complex_normalized', 'contact', 'contact_norm',
'dG_cross', 'dG_cross/dSASAx100', 'dG_separated', 'dG_separated/dSASAx100', 'dSASA_hphobic', 'dSASA_int',
'dSASA_polar', 'ddg_norep', 'ddg_rep', 'delta_unsatHbonds', 'filter_hyd_sasa', 'filter_pol_sasa',
'filter_sasa', 'h_sasa', 'hbond_E_fraction', 'hbonds_int', 'int_holes', 'int_hydcontact', 'mismatch_probability', 'nres_all',
'nres_int', 'p_sasa', 'packstat', 'per_residue_energy_int', 'pstat', 'sasa', 'sc', 'sc_value', 'side1_normalized', 'side1_score',
'side2_normalized', 'side2_score', 'ss_sc', 't_sap_score', 't_sasa', 'tb_sap_score', 'vbuns_all', 'vbuns_bb', 'vbuns_sc']]
feature_list = list(features.columns) features = np.array(features)

#Generating the training and test sets
train_features, test_features, train_labels, test_labels = train_test_split(features, labels, test_size = 0.1,
random_state = 42)
# Instantiate model with 1000 decision trees
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42, class_weight="balanced")
# Train the model on training data rf.fit(train_features, train_labels);
# Use the forest's predict method on the test data predictions = rf.predict(test_features)

y_true=[t for t in test_labels] y_pred=[int(round(p)) for p in predictions]
```

```
print('accuracy',accuracy_score(y_true, y_pred)) print('precision',precision_score(y_true, y_pred, average='micro'))
print('f1',f1_score(y_true, y_pred, average='micro')) print('recall',recall_score(y_true, y_pred, average='micro'))


#works with RF classifier import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix plot_confusion_matrix(rf, test_features, test_labels,
cmap=plt.cm.Blues)
accRandomForestClassifier76Features = accuracy_score(y_true, y_pred) from pylab import rcParams
import numpy as np
import matplotlib.pyplot as plt import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.losses import CategoricalCrossentropy from tensorflow.keras.optimizers import Adam
rel_features = rel[feature_list] # Step 1: get your data
# your features should be in a numpy array
# with shape x= number of samples x n=number of features rel_features = rel_features.to_numpy()
affinity_bins = rel['bin'].to_numpy()


nb_classes = 5
targets = affinity_bins.reshape(-1) one_hot_targets = np.eye(nb_classes)[targets]
one_hot_targets = np.delete(one_hot_targets, 0, 1)


from sklearn.model_selection import train_test_split X, y = np.asarray(rel_features).astype('float32'),
np.asarray(one_hot_targets).astype('float32')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42) feature_num = 76
n = 4


# activation function, can be sigmoid, etc act_func = 'sigmoid'


# making first layer
input = tf.keras.Input(shape=(feature_num,))


# Adding other layers (currently a 3 layer network) middle1 = tf.keras.layers.Dense(
2*feature_num,
activation=act_func, kernel_initializer=tf.keras.initializers.GlorotUniform(),
bias_initializer=tf.keras.initializers.GlorotUniform()
)(input)


#you can change the sizes middle2 = tf.keras.layers.Dense( 2*feature_num, activation='relu'
)(middle1)


#you can change the sizes middle3 = tf.keras.layers.Dense( feature_num, activation='relu'
)(middle2)


# n here is the number of bins you'd like to predict output = tf.keras.layers.Dense(
n, activation=act_func, kernel_initializer=tf.keras.initializers.GlorotUniform(),
bias_initializer=tf.keras.initializers.GlorotUniform()
)(middle3)


# building the model
model = tf.keras.Model(inputs=input, outputs=output, name='the_model') model.compile(
optimizer=tf.optimizers.Adam(learning_rate=0.1), loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
weighted_metrics=['acc'], #if that doesn't work use the one below #metrics=['accuracy'],
)


# taking a look at your model tf.keras.utils.plot_model(model, show_shapes=True)
X_train, X_test, y_train, y_test # fitting the model
model.fit(
X_train, #the train set y_train, #the train labels epochs = 10,
verbose = 1, validation_data=(
X_test, #the validation set y_test, #the validation labels
)
)


print("Evaluating model.") eval_results = model.evaluate(
X_test, #the test set y_test #the test label
)
print("Done.\n" "Test loss: {}\n" "Test accuracy: {}".format(*eval_results)) accDNN = eval_results[1]
features = df_sklearn[['side1_score', 'fa_dun_dev',
'hbonds_int', 'cms', 'dG_separated', 't_sasa',
'mismatch_probability', 'bb_sap_score']]
feature_list = list(features.columns) features = np.array(features)


#Generating the training and test sets
train_features, test_features, train_labels, test_labels = train_test_split(features, labels, test_size = 0.1,
random_state = 42)
# Instantiate model with 1000 decision trees
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42) # Train the model on training data
rf.fit(train_features, train_labels);
# Use the forest's predict method on the test data predictions = rf.predict(test_features)


y_true=[t for t in test_labels] y_pred=[int(round(p)) for p in predictions]


print('accuracy',accuracy_score(y_true, y_pred)) print('precision',precision_score(y_true, y_pred, average='micro'))
print('f1',f1_score(y_true, y_pred, average='micro')) print('recall',recall_score(y_true, y_pred, average='micro'))


#works with RF classifier import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix plot_confusion_matrix(rf, test_features, test_labels,
cmap=plt.cm.Blues)


accRandomForestClassifier8Features = accuracy_score(y_true, y_pred) get_ipython().system('pip3 install keras-
visualizer')
# first neural network with keras tutorial from numpy import loadtxt
import pandas as pd import numpy as np
from keras.models import Sequential from keras.layers import Dense import matplotlib.pyplot as plt
from keras.utils.vis_utils import plot_model from keras_visualizer import visualizer from IPython.display import Image
get_ipython().run_line_magic('matplotlib', 'inline')


# define the keras model model = Sequential()
model.add(Dense(12, input_dim=76, activation='relu')) model.add(Dense(76, activation='relu')) model.add(Dense(38,
activation='relu')) model.add(Dense(19, activation='relu')) model.add(Dense(4, activation='sigmoid'))


plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)


# compile the keras model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])


model.fit(X_train , # input data y_train , # labels epochs=150, batch_size=10)


_, accuracy = model.evaluate(X, y) print('Accuracy: %.2f' % (accuracy*100))


# make probability predictions with the model predictions = model.predict(X_test)


# your code
_, accuracy = model.evaluate(X_test, y_test) print('Accuracy: %.2f' % (accuracy*100))


accDNNsequential = accuracy


from sklearn.ensemble import GradientBoostingClassifier


clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=4,
random_state=42).fit(train_features, train_labels)
clf.score(test_features, test_labels)


# Use the forest's predict method on the test data predictions = clf.predict(test_features)
```

```
y_true=[t for t in test_labels] y_pred=[int(round(p)) for p in predictions]

print('accuracy',accuracy_score(y_true, y_pred)) print('precision',precision_score(y_true, y_pred, average='micro'))
print('f1',f1_score(y_true, y_pred, average='micro')) print('recall',recall_score(y_true, y_pred, average='micro'))

#works with RF classifier import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix plot_confusion_matrix(clf, test_features, test_labels,
cmap=plt.cm.Blues)

accGradientBoostingClassifier = accuracy_score(y_true, y_pred) from sklearn import svm
## complete the SVC set below based on the descriptions above. clf = svm.SVC(kernel= 'linear', ## add kernel
C= 100.0, ## add C
            decision_function_shape = 'ovr') ## add decision function clf.fit(train_features, train_labels.ravel())

clf.score(test_features, test_labels)

# Use the forest's predict method on the test data predictions = clf.predict(test_features)

y_true=[t for t in test_labels] y_pred=[int(round(p)) for p in predictions]

print('accuracy',accuracy_score(y_true, y_pred)) print('precision',precision_score(y_true, y_pred, average='micro'))
print('f1',f1_score(y_true, y_pred, average='micro')) print('recall',recall_score(y_true, y_pred, average='micro'))

#works with RF classifier import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix plot_confusion_matrix(clf, test_features, test_labels,
cmap=plt.cm.Blues)

accSVMlinear = accuracy_score(y_true, y_pred)

# code goes here sigma = 1
gamma = 1/(2 * sigma**2)

## Define kernel to be 'rbf'
clfg = svm.SVC(kernel= 'rbf', ## add here
             gamma=gamma, C=0.1, decision_function_shape='ovr') clfg.fit(train_features, train_labels.ravel())

clf.score(test_features, test_labels)

# Use the forest's predict method on the test data predictions = clfg.predict(test_features)

y_true=[t for t in test_labels] y_pred=[int(round(p)) for p in predictions]

print('accuracy',accuracy_score(y_true, y_pred)) print('precision',precision_score(y_true, y_pred, average='micro'))
print('f1',f1_score(y_true, y_pred, average='micro')) print('recall',recall_score(y_true, y_pred, average='micro'))

#works with RF classifier import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix plot_confusion_matrix(clfg, test_features, test_labels,
cmap=plt.cm.Blues) accSVMrbf = accuracy_score(y_true, y_pred)

allAccuracies = [accRandomForestClassifier76Features, accRandomForestClassifier8Features, accDNN,
accDNNsequential, accGradientBoostingClassifier, accSVMlinear, accSVMrbf] allAccuraciesNames =
['accRandomForestClassifier76Features', 'accRandomForestClassifier8Features', 'accDNN', 'accDNNsequential',
'accGradientBoostingClassifier', 'accSVMlinear', 'accSVMrbf'] plt.bar(allAccuraciesNames, allAccuracies)
plt.xticks(rotation=90)
```

Appendix 2

pCT-40 Vector and Proteins

Name: pCT-40 ABY
Size: 6560 bp
Resistance: Ampicillin
Use: This construct is a template for a representative affibody clone within a pCT-40 vector, suitable for plasmid production in e. coli as well as yeast surface display in EBY-100.

Summary:
Aga2--Spacer--FactorXa--HA-- PAS40--$(G_4S)_3$ Linker --NheI—Affibody(1-58)--BamHI--Myc--2Stop **174 bp region of interest**

ATGCAGTTACTTCGCTGTTTTTCAATATTTTCTGTTATTGCTTCAGTTTTAGCACAGGAACTGACAACTATATGCGAGCA
AATCCCCTCACCAACTTTAGAATCGACGCCGTACTCTTTGTCAACGACTACTATTTTGGCCAACGGGAAGGCAATGCAA
GGAGTTTTTGAATATTACAAATCAGTAACGTTTGTCAGTAATTGCGGTTCTCACCCCTCAACAACTAGCAAAGGCAGCC
CCATAAACACACAGTATGTTTTTAAGGACAATAGCTCGACGATTGAAGGTAGATACCCATACGACGTTCCAGACTACGCT
CTGCAGGCTAGTGCCTCTCCAGCTGCACCTGCTCCAGCAAGCCCTGCTGCACCAGCTCCGTCTGCTCCTGCTGCCTCT
CCAGCTGCACCTGCTCCAGCTTCTCCAGCAGCTCCTGCACCTAGTGCTCCTGCTGGGGGTGGAGGCTCTGGCGGAGG
TGGGTCTGGTGGGGGCGGATCTGCTAGCGCCGAAGCGAAATACGCTAAAGAAAACXYZAACGCGXYZXYZGAAATCXY
ZXYZCTGCCGAACCTGACCXYZXYZCAGAGAXYZGCATTCATAXYZGCACTGXYZGATGACCCGTCCCAGAGCTCTGAA
CTCCTGTCTGAGGCGAAGAAACTGAACGATTCCCAAGCACCAAAAGGATCCGAACAAAAGCTTATTTCT*GAAGAGGACT TG****TAATAG***

Sequence:
**ACGAAAGGGCCTCGTGATACGCCTATTTTTATAGGTTAATGTCATGATAATAATGGTTTCTTAGGACGGATCGCTTGCCTGTAACTTACACGCGCCTCGTA
TCTTTTAATGATGGAATAATTTGGGAATTTACTCTGTGTTTATTTATTTTTATGTTT
TGTATTTGGATTTTAGAAAGTAAATAAAGAAGGTAGAAGAGTTACGGAATGAAGAAAAAAAAATAAACAAAGGTTTAAAAAATTTCAACAAAAAGCGTAC
TTTACATATATATTTATTAGACAAGAAAAGCAGATTAAATAGATATACATTCGAT
TAACGATAAGTAAAATGTAAAATCACAGGATTTTCGTGTGTGGTCTTCTACACAGACAAGATGAAACAATTCGGCATTAATACCTGAGAGCAGGAAGAGC
AAGATAAAAGGTAGTATTTGTTGGCGATCCCCCTAGAGTCTTTTACATCTTCG
GAAAACAAAAACTATTTTTTCTTTAATTTCTTTTTTTACTTTCTATTTTTAATTTATATATTTATATTAAAAAATTTAAATTATAATTATTTTTATAGCACGTGAT
GAAAAGGACCCAGGTGGCACTTTTCGGGGAAATGTGCGCGGAACCCCTATTTG
TTTATTTTTCTAAATACATTCAAATATGTATCCGCTCATGAGACAATAACCCTGATAAATGCTTCAATAATATTGAAAAAGGAAGAGTATGAGTATTCAACA
TTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCT
CACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCAGTTGGGTGCACGAGTGGGTTACATCGAACTGGATCTCAACAGCGGTAAGATCCTTGA
GAGTTTTCGCCCCGAAGAACGTTTTCCAATGATGAGCACTTTTAAAGTTCTG
CTATGTGGCGCGGTATTATCCCGTATTGACGCCGGGCAAGAGCAACTCGGTCGCCGCATACACTATTCTCAGAATGACTTGGTTGAGTACTCACCAGTC
ACAGAAAAGCATCTTACGGATGGCATGACAGTAAGAGAATTATGCAGTGCTG
CCATAACCATGAGTGATAACACTGCGGCCAACTTACTTCTGACAACGATCGGAGGACCGAAGGAGCTAACCGCTTTTTTTCACAACATGGGGGATCATG
TAACTCGCCTTGATCGTTGGGAACCGGAGCTGAATGAAGCCATACCAAACG
ACGAGCGTGACACCACGATGCCTGTAGCAATGGCAACAACGTTGCGCAAACTATTAACTGGCGAACTACTTACTCTAGCTTCCCGGCAACAATTAATAG
ACTGGATGGAGGCGGATAAAGTTGCAGGACCACTTCTGCGCTCGGCCCTTC
CGGCTGGCTGGTTTATTGCTGATAAATCTGGAGCCGGTGAGCGTGGGTCTCGCGGTATCATTGCAGCACTGGGGCCAGATGGTAAGCCCTCCCGTATCG
TAGTTATCTACACGACGGGCAGTCAGGCAACTATGGATGAACGAAATAGAC
AGATCGCTGAGATAGGTGCCTCACTGATTAAGCATTGGTAACTGTCAGACCAAGTTTACTCATATATACTTTAGATTGATTTAAAACTTCATTTTTAATTTA
AAAGGATCTAGGTGAAGATCCTTTTTGATAATCTCATGACCAAAATCCCTTAAC
GTGAGTTTTCGTTCCACTGAGCGTCAGACCCCGTAGAAAAGATCAAAGGATCTTCTTGAGATCCTTTTTTTCTGCGCGTAATCTGCTGCTTGCAAACAAAA
AAACCACCGCTACCAGCGGTGGTTTGTTTGCCGGATCAAGAGCTACCAAC
TCTTTTTCCGAAGGTAACTGGCTTCAGCAGAGCGCAGATACCAAATACTGTCCTTCTAGTGTAGCCGTAGTTAGGCCACCACTTCAAGAACTCTGTAGCA
CCGCCTACATACCTCGCTCTGCTAATCCTGTTACCAGTGGCTGCTGCCAGTG
GCGATAAGTCGTGTCTTACCGGGTTGGACTCAAGACGATAGTTACCGGATAAGGCGCAGCGGTCGGGCTGAACGGGGGGTTCGTGCACACAGCCCAGC
TTGGAGCGAACGACCTACACCGAACTGAGATACCTACAGCGTGAGCATTG
AGAAAGCGCCACGCTTCCCGAAGGGAGAAAGGCGGACAGGTATCCGGTAAGCGGCAGGGTCGGAACAGGAGAGCGCACGAGGGAGCTTCCAGGGGG
GAACGCCTGGTATCTTTATAGTCCTGTCGGGTTTCGCCACCTCTGACTTGA
GCGTCGATTTTTGTGATGCTCGTCAGGGGGGCGGAGCCTATGGAAAAACGCCAGCAACGCGGCCTTTTTACGGTTCCTGGCCTTTTGCTGGCCTTTTGCT
CACATGTTCTTTCCTGCGTTATCCCCTGATTCTGTGGATAACCGTATTACCGC
CTTTGAGTGAGCTGATACCGCTCGCCGCAGCCGAACGACCGAGCGCAGCGAGTCAGTGAGCGAGGAAGCGGAAGAGCGCCCAATACGCAAACCGCCT
CTCCCCGCGCGTTGGCCGATTCATTAATGCAGCTGGCACGACAGGTTTCCC
GACTGGAAAGCGGGCAGTGAGCGCAACGCAATTAATGTGAGTTACCTCACTCATTAGGCACCCCAGGCTTTACACTTTATGCTTCCGGCTCCTATGTTGT
GTGGAATTGTGAGCGGATAACAATTTCACACAGGAAACAGCTATGACCATGA
TTACGCCAAGCTCGGAATTAACCCTCACTAAAGGGAACAAAAGCTGGGTACCCGACAGGTTATCAGCAACAACACAGTCATATCCATTCTCAATTAGCT
CTACCACAGTGTGTGAACCAATGTATCCAGCACCACCTGTAACCAAAACAATT
TTAGAAGTACTTTCACTTTGTAACTGAGCTGTCATTTATATTGAATTTTCAAAAATTCTTACTTTTTTTTTGGATGGACGCAAAGAAGTTTAATAATCATATT
ACATGGCATTACCACCATATACATATCCATATACATATCCATATCTAATCTTACTTA
TATGTTGTGGAAATGTAAAGAGCCCCATTATCTTAGCCTAAAAAAACCTTCTCTTTGGAACTTTCAGTAATACGCTTAACTGCTCATTGCTATATTGAAGTA
CGGATTAGAAGCCGCCGAGCGGGTGACAGCCCTCCGAAGGAAGACTCTCCT
CCGTGCGTCCTCGTCTTCACCGGTCGCGTTCCTGAAACGCAGATGTGCCTCGCGCCGCACTGCTCCGAACAATAAAGATTCTACAATACTAGCTTTTATG
GTTATGAAGAGGAAAAATTGGCAGTAACCTGGCCCCACAAACCTTCAAATG
AACGAATCAAATTAACAACCATAGGATGATAATGCGATTAGTTTTTTAGCCTTATTTCTGGGGTAATTAATCAGCGAAGCGATGATTTTTGATCTATTAACA
GATATATAAATGCAAAAAC
TGCATAACCACTTTAACTAATACTTTCAACATTTTCGGTTTGTATTACTTCTTATTCAAATGTAATAAAAGTATCAACAAAAAATTGTTAA
TATACCTCTATACTTTAACGTCAAGGAGAAAAAACTATAGAATTCTACTTCATACATTTTCAATTAAG**ATGCAGTTACTTCGCTGTTT
TTCAATATTTTCTGTTATTGCTTCAGTTTTAGCACAGGAACTGACAACTATATGCGAGCAAATCCCCTCACCAACTTTAG
AATCGACGCCGTACTCTTTGTCAACGACTACTATTTTGGCCAACGGGAAGGCAATGCAAGGAGTTTTTGAATATTACAA
ATCAGTAACGTTTGTCAGTAATTGCGGTTCTCACCCCTCAACAACTAGCAAAGGCAGCCCCATAAACACACAGTATGTT
TTTAAGGACAATAGCTCGACGATTGAAGGTAGATACCCATACGACGTTCCAGACTACGCTCTGCAGGCTAGTGCCTCTC
CAGCTGCACCTGCTCCAGCAAGCCCTGCTGCACCAGCTCCGTCTGCTCCTGCTGCCTCTCCAGCTGCACCTGCTCCAG
CTTCTCCAGCAGCTCCTGCACCTAGTGCTCCTGCTGGGGGTGGAGGCTCTGGCGGAGGTGGGTCTGGTGGGGGCGGA
TCTGCTAGCGCCGAAGCGAAATACGCTAAAGAAAACXYZAACGCGXYZXYZGAAATCXYZXYZCTGCCGAACCTGACC

XYZXYZCAGAGAXYZGCATTCATAXYZGCACTGXYZGATGACCCGTCCCAGAGCTCTGAACTCCTGTCTGAGGCGAAG
AAACTGAACGATTCCCAAGCACCAAAAGGATCCGAACAAAAGCTTATTTCT*GAAGAGGACTTGTAATAGCTCGAGAT*CTGATAACAACAGTG
TAGATGTAACAAAATCGACTTTGTTCCCACTGTACTTTTAGCTCGTACAAAATACAATATACTTTTCATTTCTCCGTAAACAACATGTTTTCCCATGTAATATCCTTTTCTATTTTT
CGTTCCGTTACCAACTTTACACATACTTTATATAGCTATT
CACTTCTATACACTAAAAAACTAAGACAATTTTAATTTTGCTGCCTGCCATATTTCAATTTGTTATAAATTCCTATAATTTATCCTATTAGTAGCTAAAAAAAGATGAATGTGAATC
GAATCCTAAGAGAATTGAGCTCCAATTCGCCCTATAGTGAGTCGTATTACAATTCACTG
GCCGTCGTTTTACAACGTCGTGACTGGGAAAACCCTGGCGTTACCCAACTTAATCGCCTTGCAGCACATCCCCCTTTCGCCAGCTGGCGTAATAGCGAAGAGGCCCGCACCGATCGC
CCTTCCCAACAGTTGCGCAGCCTGAATGGCGAATGGCGCGACGCGCCCTGTAGCGGCGC
ATTAAGCGCGGCGGGTGTGGTGGTTACGCGCAGCGTGACCGCTACACTTGCCAGCGCCCTAGCGCCCGCTCCTTTCGCTTTCTTCCCTTCCTTTCTCGCCACGTTCGCCGGCTTTCC
CCGTCAAGCTCTAAATCGGGGGCTCCCTTTAGGGTTCCGATTTAGTGCTTTACGGCACC
TCGACCCCAAAAAACTTGATTAGGGTGATGGTTCACGTAGTGGGCCATCGCCCTGATAGACGGTTTTTCGCCCTTTGACGTTGGAGTCCACGTTCTTTAATAGTGGACTCTTGTTCC
AAACTGGAACAACACTCAACCCTATCTCGGTCTATTCTTTTGATTTATAAGGGATTTTG
CCGATTTCGGCCTATTGGTTAAAAAATGAGCTGATTTAACAAAAATTTAACGCGAATTTTAACAAAATATTAACGTTTACAATTTCCTGATGCGGTATTTTCTCCTTACGCATCTGT
GCGGTATTTCACACCGCAGGCAAGTGCACAAACAATACTTAAATAAATACTACTCAGTA
ATAACCTATTTCTTAGCATTTTTGACGAAATTTGCTATTTTGTTAGAGTCTTTTACACCATTTGTCTCCACACCTCCGCTTACATCAACACCAATAACGCCATTTAATCTAAGCGCA
TCACCAACATTTTCTGGCGTCAGTCCACCAGCTAACATAAAATGTAAGCTTTCGGGGCT
CTCTTGCCTTCCAACCCAGTCAGAAATCGAGTTCCAATCCAAAAGTTCACCTGTCCCACCTGCTTCTGAATCAAACAAGGGAATAAACGAATGAGGTTTCTGTGAAGCTGCACTGAG
TAGTATGTTGCAGTCTTTTGGAAATACGAGTCTTTTAATAACTGGCAAACCGAGGAACT
CTTGGTATTCTTGCCACGACTCATCTCCATGCAGTTGGACGATATCAATGCCGTAATCATTGACCAGAGCCAAAACATCCTCCTTAGGTTGATTACGAAACACGCCAACCAAGTATT
TCGGAGTGCCTGAACTATTTTTATATGCTTTTACAAGACTTGAAATTTTCCTTGCAATA
ACCGGGTCAATTGTTCTCTTTCTATTGGGCACACATATAATACCCAGCAAGTCAGCATCGGAATCTAGAGCACATTCTGCGGCCTCTGTGCTCTGCAAGCCGCAAACTTTCACCAAT
GGACCAGAACTACCTGTGAAATTAATAACAGACATACTCCAAGCTGCCTTTGTGTGCTT
AATCACGTATACTCACGTGCTCAATAGTCACCAATGCCCTCCCTCTTGGCCCTCTCCTTTTCTTTTTTCGACCGAATTAATTCTTAATCGGCAAAAAAAGAAAAGCTCCGGATCAAG
ATTGTACGTAAGGTGACAAGCTATTTTTCAATAAAGAATATCTTCCACTACTGCCATCT
GGCGTCATAACTGCAAAGTACACATATATTACGATGCTGTCTATTAAATGCTTCCTATATTATATATATATAGTAATGTCGTTTATGGTGCACTCTCAGTACAATCTGCTCTGATGCCG
CATAGTTAAGCCAGCCCCGACACCCGCCAACACCCGCTGACGCGCCCTGACGGGCTTGT
CTGCTCCCGGCATCCGCTTACAGACAAGCTGTGACCGTCTCCGGGAGCTGCATGTGTCAGAGGTTTTCACCGTCATCACCGAAACGCGCGA

Protein:
Aga2p – KDNSST – Xa – HA – PAS40–$(G_4S)_3$ – AS – Affi(1 - 58)– GS – c-myc
MQLLRCFSIFSVIASVLAQELTTICEQIPSPTLESTPYSLSTTTILANGKAMQGVFEYYKSVTFVSNCGSHP
STTSKGSPINTQYVFKDNSSTIEGRYPYDVPDYALQAS
ASPAAPAPASPAAPAPSAPAASPAAPAPASPAAPAPSAPAGGGGSGGGGSGGGGSASAEAKYAKENxN
AFxEIxxLPNLNxxQRxAFIxALxDDPSQSSELLSEAKKLNDSQAPKGSEQKLISEEDL

Affibody Sanger sequence results

G1C1:ANNNNNNNNNNNNNNNACATGGGAAACATGTTGTTTACGGAGAAATGA
AAAGTATATTGTATTTTGTACGAGCTAAAAGTACAGTGGGAACAAAGTCGATT
TTGTTACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTCTTCA
GAAATAAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTTTCTT
CGCCTCAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTCAAGCAGTGCCTC
AAAGAATGCCGCTTTCTGGTCGCCGGTCAGGTTCGGCAGGACCCAGATGGAA
TTGGCCGCGGCAGTCACTTCTTTGTTGTATTTCGCTTCGGCGCTAGCAGATCC
GCCCCCACCAGACCCACCTCCGCCAGAGCCTCCACCCCCAGCAGGAGCACT
AGGTGCAGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTGGAGAGGCAG
CAGGAGCAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCAGGTGCAGCT
GGAGAGGCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTATGGGTATCTAC
CTTCAATCGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTTATGGGGCTG
CCTTTGCTANNNNNTGAGGGA
G1C2:NNNNNNNNNNNNNNNNACNTGGNANNNTGTTGTTTACGGAGAAATGA
AAAGTATATTGTATTTTGTACGAGCTAAAAGTACNGTGGGAACAAAGTCGATT
TTGTTACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTCTTCA
GAAATAAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTTTCTT
CGCCTCAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTCGAGCAGTGCAGC
ATAGAATGCGGCTTTCTGGAAGGCGGTCAGGTTCGGCAGATACCGGATCTGG
GTGGCCGCGGAGGACACTTCTTTGTAGTATTTCGCTTCGGCGCTAGCAGATCC
GCCCCCACCAGACCCACCTCCGCCAGAGCCTCCACCCCCAGCAGGAGCACT
AGGTGCAGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTGGAGAGGCAG
CAGGAGCAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCAGGTGCAGCT
GGAGAGGCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTATGGGTATCTAC
CTTCAATCGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTTATGGGGCTG
CCTTTGCTAGNTGTTGAGGGA
G1C3:NNNNNNNNTTANNTGGGAAACATGTTGTTTACGGAGAAATGAAAAGT
ATATTGTATTTTGTACGAGCTAAAAGTACAGTGGGAACAAAGTCGATTTTGTT
ACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTCTTCAGAAAT
AAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTTTCTTCGCCT
CAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTTAAACAGTGCGTAGACGA
ATGCATTTCTCTGGTCGAGGGTCAGGTTCGGCAGATTATCGATCACATCGTAC
GCGTCAAACCCTTCTTTGGTGTATTTCGCTTCGGCGCTAGCAGATCCGCCCCC

ACCAGACCCACCTCCGCCAGAGCCTCCACCCCCAGCAGGAGCACTAGGTGC
AGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTGGAGAGGCAGCAGGAG
CAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCAGGTGCAGCTGGAGAG
GCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTATGGGTATCTACCTTCAAT
CGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTTATGGGGCTGCCTTNNN
NNNNNNNNNNNNNAGGGGNNNNNN
G2C1:ANNNNNNNNNNNNNNNNNNNCNTGGGAAACATGTTGTTTACGGAGAAA
TGAAAAGTATATTGTATTTTGTACGAGCTAAAAGTACNGTGGGAACAAAGTC
GATTTTGTTACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTC
TTCAGAAATAAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTT
TCTTCGCCTCAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTTCCCCAGTGC
AGCGATGAATGCACGTATCTGATAGCCGGTCAGGTTCGGCAGGGACCGGATG
GACATGTCCGCGTCCAACCATTCTTTGTAGTATTTTGCTTCGGCGCTAGCAGA
ACCACCACCACCAGACCCACCTCCGCCAGAGCCTCCACCCCCAGCAGGAGC
ACTAGGTGCAGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTGGAGAGG
CAGCAGGAGCAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCAGGTGCA
GCTGGAGAGGCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTATGGGTATC
TACCTTCAATCGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTTATGGGG
CTGCCTTTGCTAGTTGTTGAGGGNNN
G2C2:NNNNNNNNNNTTANNTGGGANACATGTTGTTTACGGAGAAATGAAAA
GTATATTGTATTTTGTACGAGCTAAAAGTACAGTGGGAACAAAGTCGATTTTG
TTACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTCTTCAGAA
ATAAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTTTCTTCGC
CTCAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTTAGACAGTGCGTCGAC
GAATGCGGCTTTCTGAGACGCGGTCAGGTTCGGAGGTAGAAGATCACAACGT
ACGCGTTAAACTGTTCTTTAGCGTATTTCGCTTCGGCGCTAGCAGATCCGCCC
CCACCAGACCCACCTCCGCCAGAGCCTCCACCCCCAGCAGGAGCACTAGGT
GCAGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTGGAGAGGCAGCAGG
AGCAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCAGGTGCAGCTGGAG
AGGCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTATGGGTATCTACCTTC
AATCGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTTATGGGGCTGCCTT
TGCTANNNNNNTTGAGGGGA
G2C3:NNNNGNNNNTACNTGGGAAACATGTTGTTTACGGAGAAATGAAAAGT
ATATTGTATTTTGTACGAGCTAAAAGTACAGTGGGAACAAAGTCGATTTTGTT
ACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTCTTCAGAAAT
AAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTTTCTTCGCCT
CAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTCATCCAGTGCCTCGATGAA
TGCGCCTTTCTGAACGCGGGTCAGGTTCGGCAGATAAAAGATCACCACGGCC
GCGGCGCGCCTTTCTTTGTAGTATTTTGCTTCGGCGCTAGCAGAACCACCACC
ACCAGAACCACCACCACCAGAACCACCTCCGCCAGAGCCTCCACCCCCAGC
AGGAGCACTAGGTGCAGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTG
GAGAGGCAGCAGGAGCAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCA
GGTGCAGCTGGAGAGGCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTAT
GGGTATCTACCTTCAATCGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTT
ATGGGGCTGCCTTNNNNNNNNNNNTNNNNNNNNGGGGNNNAA
G3C1:NNNNNNNNNNNNNNNNNACNTGGGAANCATGTTGTTTACGGAGAAATG
AAAAGTATATTGTATTTTGTACGAGCTAAAAGTACAGTGGGAACAAAGTCGA
TTTTGTTACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTCTT
CAGAAATAAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTTTC
TTCGCCTCAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTTATACAGTGCCC
GCGCGAATGCGACTCTCTGACCCTGGGTCAGGTTCGGCAGATCATTGATCAC
ATAGATCGCGTTCAACCTTTCTTTAGCGTATTTCGCTTCGGCGCTAGCAGATC
CGCCCCCACCAGACCCACCTCCGCCAGAGCCTCCACCCCCAGCAGGAGCAC
TAGGTGCAGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTGGAGAGGCA
GCAGGAGCAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCAGGTGCAGC
TGGAGAGGCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTATGGGTATCTA
CCTTCAATCGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTTATGGGGCT
GCCTTTGCTANNNGTTGAGGGAN
G3C2:NNNNNGNNTTANNTGGGAAACATGTTGTTTACGGAGAAATGAAAAGT
ATATTGTATTTTGTACGAGCTAAAAGTACAGTGGGAACAAAGTCGATTTTGTT
ACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTCTTCAGAAAT

AAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTTTCTTCGCCT
CAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTTAGACAGTGCGTCGACGA
ATGCGGCTTTCTGAGACGCGGTCAGGTTCGGCAGGTAGAAGATCACAACGTA
CGCGTTAAACTGTTCTTTAGCGTATTTCGCTTCGGCGCTAGCAGATCCGCCCC
CACCAGACCCACCTCCGCCAGAGCCTCCACCCCCAGCAGGAGCACTAGGTG
CAGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTGGAGAGGCAGCAGGA
GCAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCAGGTGCAGCTGGAGA
GGCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTATGGGTATCTACCTTCAA
TCGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTTATGGGGCTGCCTNNN
NNNNNNNTNNTNNGNNNGGNNNNNAAA
G3C3:NNNNNNNNNTACNTGGGAANCATGTTGTTTACGGAGAAATGAAAAGT
ATATTGTATTTTGTACGAGCTAAAAGTACAGTGGGAACAAAGTCGATTTTGTT
ACATCTACACTGTTGTTATCAGATCTCGAGCTATTACAAGTCCTCTTCAGAAAT
AAGCTTTTGTTCGGATCCTTTTGGTGCTTGGGAATCGTTCAGTTTCTTCGCCT
CAGACAGGAGTTCAGAGCTCTGGGACGGGTCGTTCCCCAGTGCAGCGATGA
ATGCACGTATCTGATAGCCGGTCAGGTTCGGCAGGGACCGGATGGACATGTC
CGCGTCCAACCATTCTTTGTAGTATTTTGCTTCGGCGCTAGCAGAACCACCAC
CACCAGACCCACCTCCGCCAGAGCCTCCACCCCCAGCAGGAGCACTAGGTG
CAGGAGCTGCTGGAGAAGCTGGAGCAGGTGCAGCTGGAGAGGCAGCAGGA
GCAGACGGAGCTGGTGCAGCAGGGCTTGCTGGAGCAGGTGCAGCTGGAGA
GGCACTAGCCTGCAGAGCGTAGTCTGGAACGTCGTATGGGTATCTACCTTCAA
TCGTCGAGCTATTGTCCTTAAAAACATACTGTGTGTTTATGGGGCTGCCTTTG
CTANNNNGTTGAGGGGN

Affibody mutant g-block sequences

G3C1 Affibody (wild-type): GGSASAEAKYAKERLNAIYVINDLPNLTQGQRVAFARALYNDPSQSSELLSEAK
KLNDSQAPKGSEQKLISEE

Forward sequence: 5'- GGG GCG GAT CTG CTA GCG CCG AAG CGA AAT ACG CTA AAG AAA GGT TGA
ACG CGA TCT ATG TGA TCA ATG ATC TGC CGA ACC TGA CCC AGG GTC AGA GAG TCG CAT TCG
CGC GGG CAC TGT ATA ACG ACC CGT CCC AGA GCT CTG AAC TCC TGT CTG AGG CGA AGA AAC
TGA ACG ATT CCC AAG CAC CAA AAG GAT CCG AAC AAA AGC TTA TTT CTG AAG AG -3'
Reverse sequence: 5'- CTC TTC AGA AAT AAG CTT TTG TTC GGA TCC TTT TGG TGC TTG GGA ATC GTT
CAG TTT CTT CGC CTC AGA CAG GAG TTC AGA GCT CTG GGA CGG GTC GTT ATA CAG TGC CCG
CGC GAA TGC GAC TCT CTG ACC CTG GGT CAG GTT CGG CAG ATC ATT GAT CAC ATA GAT CGC
GTT CAA CCT TTC TTT AGC GTA TTT CGC TTC GGC GCT AGC AGA TCC GCC CC
-3'
Total length: 221 Total Tm: 68.8 ºC Total GC%: 52
F_overlap length: 17 F_overlap Tm: 58.6 ºC R_overlap length: 30 R_overlap Tm: 57.8 ºC

V28W: GGSASAEAKYAKERLNAIYVINDLPNLTQGQRWAFARALYNDPSQSSELLSEAK
KLNDSQAPKGSEQKLISEE

Forward sequence: 5'- GGG GCG GAT CTG CTA GCG CCG AAG CGA AAT ACG CTA AAG AAA GGT TGA
ACG CGA TCT ATG TGA TCA ATG ATC TGC CGA ACC TGA CCC AGG GTC AGA GAT GGG CAT TCG
CGC GGG CAC TGT ATA ACG ACC CGT CCC AGA GCT CTG AAC TCC TGT CTG AGG CGA AGA AAC
TGA ACG ATT CCC AAG CAC CAA AAG GAT CCG AAC AAA AGC TTA TTT CTG AAG AG -3'
Reverse sequence: 5'- CTC TTC AGA AAT AAG CTT TTG TTC GGA TCC TTT TGG TGC TTG GGA ATC GTT
CAG TTT CTT CGC CTC AGA CAG GAG TTC AGA GCT CTG GGA CGG GTC GTT ATA CAG TGC CCG
CGC GAA TGC CCA TCT CTG ACC CTG GGT CAG GTT CGG CAG ATC ATT GAT CAC ATA GAT CGC
GTT CAA CCT TTC TTT AGC GTA TTT CGC TTC GGC GCT AGC AGA TCC GCC CC
-3'
Total length: 221 Total Tm: 68.9 ºC Total GC%: 52
F_overlap length: 17 F_overlap Tm: 58.6 ºC R_overlap length: 30 R_overlap Tm: 57.8 ºC

V28T: GGSASAEAKYAKERLNAIYVINDLPNLTQGQRTAFARALYNDPSQSSELLSEAK
KLNDSQAPKGSEQKLISEE

Forward sequence: 5'- GGG GCG GAT CTG CTA GCG CCG AAG CGA AAT ACG CTA AAG AAA GGT TGA ACG CGA TCT ATG TGA TCA ATG ATC TGC CGA ACC TGA CCC AGG GTC AGA GAA CCG CAT TCG CGC GGG CAC TGT ATA ACG ACC CGT CCC AGA GCT CTG AAC TCC TGT CTG AGG CGA AGA AAC TGA ACG ATT CCC AAG CAC CAA AAG GAT CCG AAC AAA AGC TTA TTT CTG AAG AG -3'
Reverse sequence: 5'- CTC TTC AGA AAT AAG CTT TTG TTC GGA TCC TTT TGG TGC TTG GGA ATC GTT CAG TTT CTT CGC CTC AGA CAG GAG TTC AGA GCT CTG GGA CGG GTC GTT ATA CAG TGC CCG CGC GAA TGC GGT TCT CTG ACC CTG GGT CAG GTT CGG CAG ATC ATT GAT CAC ATA GAT CGC GTT CAA CCT TTC TTT AGC GTA TTT CGC TTC GGC GCT AGC AGA TCC GCC CC
-3'
Total length: 221 Total Tm: 68.9 ºC Total GC%: 52
F_overlap length: 17 F_overlap Tm: 58.6 ºC R_overlap length: 30 R_overlap Tm: 57.8 ºC

V28D: GGSASAEAKYAKERLNAIYVINDLPNLTQGQRDAFARALYNDPSQSSELLSEAK KLNDSQAPKGSEQKLISEE
Forward sequence: 5'- GGG GCG GAT CTG CTA GCG CCG AAG CGA AAT ACG CTA AAG AAA GGT TGA ACG CGA TCT ATG TGA TCA ATG ATC TGC CGA ACC TGA CCC AGG GTC AGA GAG ACG CAT TCG CGC GGG CAC TGT ATA ACG ACC CGT CCC AGA GCT CTG AAC TCC TGT CTG AGG CGA AGA AAC TGA ACG ATT CCC AAG CAC CAA AAG GAT CCG AAC AAA AGC TTA TTT CTG AAG AG -3'
Reverse sequence: 5'- CTC TTC AGA AAT AAG CTT TTG TTC GGA TCC TTT TGG TGC TTG GGA ATC GTT CAG TTT CTT CGC CTC AGA CAG GAG TTC AGA GCT CTG GGA CGG GTC GTT ATA CAG TGC CCG CGC GAA TGC GTC TCT CTG ACC CTG GGT CAG GTT CGG CAG ATC ATT GAT CAC ATA GAT CGC GTT CAA CCT TTC TTT AGC GTA TTT CGC TTC GGC GCT AGC AGA TCC GCC CC
-3'
Total length: 221 Total Tm: 68.8 ºC Total GC%: 52
F_overlap length: 17 F_overlap Tm: 58.6 ºC R_overlap length: 30 R_overlap Tm: 57.8 ºC

A33Q: GGSASAEAKYAKERLNAIYVINDLPNLTQGQRVAFARQLYNDPSQSSELLSEAK KLNDSQAPKGSEQKLISEE

Forward sequence: 5'- GGG GCG GAT CTG CTA GCG CCG AAG CGA AAT ACG CTA AAG AAA GGT TGA ACG CGA TCT ATG TGA TCA ATG ATC TGC CGA ACC TGA CCC AGG GTC AGA GAG TCG CAT TCG CGC GGC AGC TGT ATA ACG ACC CGT CCC AGA GCT CTG AAC TCC TGT CTG AGG CGA AGA AAC TGA ACG ATT CCC AAG CAC CAA AAG GAT CCG AAC AAA AGC TTA TTT CTG AAG AG -3'
Reverse sequence: 5'- CTC TTC AGA AAT AAG CTT TTG TTC GGA TCC TTT TGG TGC TTG GGA ATC GTT CAG TTT CTT CGC CTC AGA CAG GAG TTC AGA GCT CTG GGA CGG GTC GTT ATA CAG CTG CCG CGC GAA TGC GAC TCT CTG ACC CTG GGT CAG GTT CGG CAG ATC ATT GAT CAC ATA GAT CGC GTT CAA CCT TTC TTT AGC GTA TTT CGC TTC GGC GCT AGC AGA TCC GCC CC
-3'
Total length: 221 Total Tm: 68.8 ºC Total GC%: 52
F_overlap length: 17 F_overlap Tm: 58.6 ºC R_overlap length: 30 R_overlap Tm: 57.8 ºC

A33L: GGSASAEAKYAKERLNAIYVINDLPNLTQGQRVAFARLLYNDPSQSSELLSEAKK LNDSQAPKGSEQKLISEE
Forward sequence: 5'- GGG GCG GAT CTG CTA GCG CCG AAG CGA AAT ACG CTA AAG AAA GGT TGA ACG CGA TCT ATG TGA TCA ATG ATC TGC CGA ACC TGA CCC AGG GTC AGA GAG TCG CAT TCG CGC GGT TGC TGT ATA ACG ACC CGT CCC AGA GCT CTG AAC TCC TGT CTG AGG CGA AGA AAC TGA ACG ATT CCC AAG CAC CAA AAG GAT CCG AAC AAA AGC TTA TTT CTG AAG AG -3'
Reverse sequence: 5'- CTC TTC AGA AAT AAG CTT TTG TTC GGA TCC TTT TGG TGC TTG GGA ATC GTT CAG TTT CTT CGC CTC AGA CAG GAG TTC AGA GCT CTG GGA CGG GTC GTT ATA CAG CAA CCG CGC GAA TGC GAC TCT CTG ACC CTG GGT CAG GTT CGG CAG ATC ATT GAT CAC ATA GAT CGC GTT CAA CCT TTC TTT AGC GTA TTT CGC TTC GGC GCT AGC AGA TCC GCC CC
-3'
Total length: 221 Total Tm: 68.9 ºC Total GC%: 51.6
F_overlap length: 17 F_overlap Tm: 58.6 ºC R_overlap length: 30 R_overlap Tm: 57.8 ºC

A33Y: GGSASAEAKYAKERLNAIYVINDLPNLTQGQRVAFARYLYNDPSQSSELLSEAK KLNDSQAPKGSEQKLISEE

Forward sequence: 5'- GGG GCG GAT CTG CTA GCG CCG AAG CGA AAT ACG CTA AAG AAA GGT TGA ACG CGA TCT ATG TGA TCA ATG ATC TGC CGA ACC TGA CCC AGG GTC AGA GAG TCG CAT TCG CGC GGT ACC TGT ATA ACG ACC CGT CCC AGA GCT CTG AAC TCC TGT CTG AGG CGA AGA AAC TGA ACG ATT CCC AAG CAC CAA AAG GAT CCG AAC AAA AGC TTA TTT CTG AAG AG -3'
Reverse sequence: 5'- CTC TTC AGA AAT AAG CTT TTG TTC GGA TCC TTT TGG TGC TTG GGA ATC GTT CAG TTT CTT CGC CTC AGA CAG GAG TTC AGA GCT CTG GGA CGG GTC GTT ATA CAG GTA CCG CGC GAA TGC GAC TCT CTG ACC CTG GGT CAG GTT CGG CAG ATC ATT GAT CAC ATA GAT CGC GTT CAA CCT TTC TTT AGC GTA TTT CGC TTC GGC GCT AGC AGA TCC GCC CC
-3'
Total length: 221 Total Tm: 68.8 ºC Total GC%: 51.6
F_overlap length: 17 F_overlap Tm: 58.6 ºC R_overlap length: 30
R_overlap Tm: 57.8 ºC

References

Wang J, Leung KS, Chow SKH, Cheung WH (2017). The effect of whole body vibration on fracture healing-a systematic review. European Cells and Materials 34:108-127.

Dorogin J, Townsend JM, Hettiaratchi MH. 2021. Biomaterials for protein delivery for complex tissue healing responses. Biomaterial Science. 9: 2339-2361.

Miyazono K, Kamiya Y, Morikama M. 2010. Bone morphogenic receptors and signal transduction. Journal of Biochemistry. 147(1): 35-51.

Hettiaratchi M, Laxminarayanan K, Rouse T, Chou C, McDevitt TC, Guldberg RE. 2020. Heparin-mediated delivery of bone morphogenetic protein-2 improves spatial localization of bone regeneration. Science Advances. 6: eaay1240.

Angelini A, Chen TF, Picciotto SD, Yang NJ, Tzeng A, Santos MS, Van Deventer JA, Traxlmayr MW, Wittrup DK. 2015. Protein Engineering and Selection Using Yeast Surface Display. In: Lui B, editor. Yeast Surface Display: Methods, Protocols, and Applications, Methods in Molecular Biology, vol. 1319. New York (US): Springer Science+Business Media. p. 3-36.

Anfinsen, Haber, Sela, White. 1961. The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. National Academy of Sciences. 47(9): 1309-1314.

Dill KA, Chan HS. 1997. From Levinthal to pathways to funnels. Nature publishing group. 4(1): 10-19.

Huang PS, Boyken S, Baker D. 2016. The coming of age of de novo protein design. Nature. 537: 320-327.

Rocklin GJ, Chidyausika TM, Goreshnik I, Ford A, Houliston S, Lemak A, Carter L, Ravichandran R, Mulligan VK, Chevalier A, et al. 2017. Global analysis of protein folding using massively parallel design, synthesis, and testing. Science. 357(6347): 168-175.

Chevalier A, Silva DA, Rocklin GJ, Hicks DR, Vergara R, Murapa P, Bernard SM, Zhang L, Lam K, Yao G, et al. 2017. Massively parallel de novo protein design for targeted therapeutics. Nature. 550: 74-79.

Linsky T , Noble K, Tobin A, Crow R, Carter L, Urbauer J, Baker D, Strauch EM. 2021. Sampling of structure and sequence space of small protein folds. BioRxviv.

Leaver-Fay A, Tyka M, Lewis SM, Lange OL, Thompson J, Jacak R, Kaufman KW, Renfrew PD, Smith CA, Sheffler W, et al. 2011. Rosetta3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. In: Michael L. Johnson, Ludwig Brand, editors. Methods in Enzymology, vol. 487. Cambridge (MA). Academic Press. p. 545-574.

Silva DA, Correia BE, Procko E. 2016. Motif-Driven Design of Protein-Protein Interfaces. In: Stoffard BL, editor. Computational Design of Ligand Binding Proteins, Methods in Molecular Biology, vol. 1414. New York (US): Springer Science+Business Media. p. 285-304.

Coventry B. 2021. Learning how to make mini-proteins that bind to specific target proteins [thesis]. [Seattle (WA)]: University of Washington.

Cao L, Coventry B, Goreshnik I, Huang B, Park JS, Jude KM, Marković I, Kadam RU, Verschueren KHG, Verstraete K, et al. 2021. Robust de novo design of protein binding proteins from target structural information alone. BioRxviv.

Jendrusch M, Korbel JO, Sadiq SK. 2021. AlphaDesign: a de novo protein design framework based on AlphaFold.